ESD-TR-66- 330

MTR-285

FINAL REPORT - JOINT AFLC/ESD/MITRE ADVANCED DATA MANAGEMENT (ADAM) EXPERIMENT

FEBRUARY 1967

B. F. Char

A. C. Foreman

Prepared for

DEPUTY FOR ENGINEERING AND TECHNOLOGY
COMPUTER PROGRAMMING DIVISION

ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

Project 503F

Prepared by

THE MITRE CORPORATION
Bedford, Massachusetts

Contract AF19(628)-5165

ESD-TR-66- 330

MTR-285

FINAL REPORT - JOINT AFLC/ESD/MITRE ADVANCED DATA MANAGEMENT (ADAM) EXPERIMENT

FEBRUARY 1967

B. F. Char

A. C. Foreman

Prepared for

DEPUTY FOR ENGINEERING AND TECHNOLOGY
COMPUTER PROGRAMMING DIVISION

ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

Project 503F

Prepared by

THE MITRE CORPORATION
Bedford, Massachusetts

Contract AF19(628)-5165

FOREWORD

The authors wish to express their appreciation for the support given to the experiment by the Air Force Logistics Command (AFLC) ADAM Users Group. In particular, we wish to acknowledge the contribution of the Section V, the part including User Reactions, by Donald Simmons, the AFLC Project Monitor.

The authors also wish to thank Otto Beebe and John Penney of the MITRE Corporation's Information Processing Department for their many contributions to this report.

REVIEW AND APPROVAL

This technical report has been reviewed and is approved.

Paul G. Galentine Jr.

PAUL G. GALENTINE, JR.
Colonel, USAF
Deputy for Command Systems

ABSTRACT

This final report describes the components of the Joint AFLC/ESD/MITRE Advanced Data Management Experiment (ADAM) and the process of implementation. The objective of the experiment was to determine the applicability of Generalized Data Management Systems such as ADAM to management information problems as found in AFLC. Observations concerning this applicability are given from two user viewpoints: programmer-user and the application or mission-oriented user.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION I

INTRODUCTION

BACKGROUND INFORMATION

One of the problems in any rapidly advancing technological field is translating concepts developed in a laboratory environment into a useful operational application. This is especially true in computer programming technology. The purpose of the project described in this report is to bridge the gap between concept development and application for one such programming concept, Generalized Data Management.

The generalized data management concept is a result of the observation that many of the functions performed by command or management information systems are common to a large majority of the systems of that class. Thus, if routines can be written that are general enough to perform these common functions, then the programmer can concentrate almost exclusively on the functions that are peculiar to his problem. One method of obtaining generality is to keep data completely divorced from the routines, and treat all problem-peculiar aspects of the function as data. More background material concerning the concept of using a computer to reduce the complexity of the computer programmer's task is contained in Appendix II, for the benefit of the non-programmer.

Several systems based on the generalized data management concept have been built for use in a laboratory. For example, the Advanced Data Management (ADAM) System was designed and programmed by The MITRE Corporation for use as an information system design tool.

More sophisticated systems based on the generalized data management concept and other ideas have been proposed, but have not received approval. A major criticism of these proposed systems has been the lack of a firm technical foundation. Although ADAM and other generalized data management systems were nearing completion when this project was proposed, they had not been tested on a problem from the field.

Because the generalized data management concept had not been field-tested, the Joint ESD/MITRE/AFLC Advanced Data Management Experiment was proposed with the following objectives:

(a) to demonstrate the generalized data management concept and isolate specific areas for improvement;

(b) to educate users in the potential of that concept; and

(c) to provide AFLC with a testbed on which to try out new system and policy level ideas prior to consideration for implementation in the field.

The approach taken to accomplish these objectives was to select an existing management information system and reprogram it on an ADAM base. The ADAM system was selected from generalized data management systems because it and its developers were available to the Electronic Systems Division (ESD) in the ESD/MITRE Systems Design Laboratory.

The Air Force Logistics Command (AFLC) was asked to join ESD in the experiment for several reasons:

(a) AFLC is the largest user of electronic data processing in the federal government.

(b) ESD had established a good working relationship with several offices at AFLC through previous work; and

(c) AFLC had several large scale management information systems that were technically suitable for the experiment.

Meetings were held at HQ AFLC during December 1964 and January 1965 to select a data system to use as a problem for the test and to plan the experiment. Representatives of the Electronic Systems Division, The MITRE Corporation, and the Logistics Command participated in the discussions.

The Category I and IIR Consumption Item Requirements Computation (D041) System, described later, was selected for the test. Based on the information gathered at these meetings, a plan was prepared by ESD and MITRE and presented to Major General J. W. O'Neill, Commander, Electronic Systems Division. General O'Neill approved the project and work began on 1 April 1965. More information concerning the initial plans for the experiment is contained in Reference 1.

The initial portion of the project was primarily concerned with learning about the D041 system. Documentation on the system was supplemented by a complete set of the D041 master file magnetic tapes. These tapes, received in June 1965, were analyzed to determine the information necessary to design the file structures for the ADAM-based Requirements System. This data base analysis activity was followed by a detailed study of the procedures used in the computational portion of the D041 system. This effort was aided by obtaining a set of input, intermediate and output computer tapes in August 1965 for studying the relationships between input and output data.

Based on these two study efforts, a program design was formulated, and work began to construct an ADAM-based Requirements Computation System.

An initial version of the ADAM-based Requirements System was completed early in December 1965. This initial system contained very few of the procedures used to compute requirements and a small subset of the data base. The information retrieval capability was available via the FABLE query language. Six members of the AFLC-ADAM users group visited the Systems Design Laboratory to exercise the initial ADAM-based Requirements System and offer suggestions for future versions.

An updated data base (from the FY 67-1 requirements computation) was received in April 1966. Several new subsets of the data base were generated for experimentation during the period of use by the AFLC users group.

On 1 April 1966, a remote query station consisting of a teletype and high-speed printer was installed at HQ AFLC, Wright-Patterson AFB, Ohio, so that more AFLC personnel could make use of the system in an on-line mode of operation. AFLC personnel were trained in the use of the remote station and the FABLE query language so that they could use the system directly. MITRE personnel assisted in formulating the more complex queries. The reactions of the users of the system are detailed in Section V. The remote station remained at HQ AFLC through August 1966. The experiment was completed on 31 August 1966.

DESCRIPTION OF THE ADAM SYSTEM

Before proceeding further, an explanation of the ADAM system is in order. ADAM is an integrated set of generalized computer programs designed to perform common and management data processing functions. It was implemented by The MITRE Corporation on the IBM 7030 (STRETCH) computer in the ESD/MITRE Systems Design Laboratory. It is a design tool to be used by management information system designers to construct functional prototypes of proposed information systems so that tests may be performed before writing the specifications for the operational system.

A few of the common functions provided for in the ADAM system are:

(a) Data base generation - describing and entering data into the system;

(b) Data base searching and analysis - retrieving data from the system and performing computations with it;

(c) Input message processing - recognizing and translating interpretively messages entered on-line and controlling the performance of the tasks described by the message;

(d) Report generating, formatting, and display-controlling the output of information to the appropriate typewriter, printer, or cathode-ray tube display in the appropriate format; and

(e) Providing for special processing routines - interfacing with problem-specific routines written in assembly or compiler languages.

All data and problem specific information are stored in the computer's memory using a hierarchical ordering scheme known as a file, and are completely divorced from the programs. Related entities called objects are grouped together in files and are described by properties.

The concept of generality is extended to the translator itself. New languages may be introduced to the ADAM system by describing the syntax and semantics (the rules for translation) of the language to the translator. At present, several languages have been described to the translator. The FABLE language provides a data retrieval and computation capability. The Initial File Generation Language (IFGL) allows the programmer to incorporate data bases into the ADAM system. Assembly language routines may be added by providing interface instructions using the DAMSEL compiler language. A restricted set of FORTRAN statements may be used to construct routines if the COMFORT postprocessor is used to provide the link to the ADAM system.

The ADAM system is designed to operate most effectively in an on-line mode. This implies that the necessary hardware (printers, input typewriters, cathode-ray tube displays, etc.) are available. Appendix I contains a comparison of the hardware configuration of the IBM 7030 at MITRE and of the IBM 7080 used by the Air Force Logistics Command. More information concerning the ADAM System itself may be obtained by consulting the list of references.

SELECTION OF DATA SYSTEM

Now that the ADAM System has been discussed, the AFLC data system that was selected as a test application will be explained.

The Category I and IIR Consumption Item Requirements Computation (D041) System develops information for Logistics Command managers concerning the quantities of spare parts required to support Air Force weapon systems during the next five years. These forecasted quantities are developed from historical usage data and the projected use of the weapon systems. The requirements thus computed are balanced against existing and projected assets of the stock item to determine if the Air Force will be over- or understocked. Buy orders may be initiated or repair schedules adjusted if an understocked position is indicated. On the other hand, procurement contracts may be terminated in situations where the Air Force will be overstocked in a particular item. The costs of buy requirements and the dollar value of projected stock overages are reported by the system.

D041 System was programmed at HQ AFLC for the IBM 7080 computer. The complete program is run every three months at each of the Air Material Areas (AMAs) for the Category I and IIR items (items which may be returned to a depot for repair) in the inventory, and each AMA manages between 2,000 and 20,000 of them. Each item has an average of approximately 2,000 characters of information associated with it in the data base for a total of 170 million characters. The D041 system programs contain a total of approximately 125,000 instructions segmented into many different runs. The system is run in a serial batch-processing mode using 20 high-speed tape drives. It requires approximately 12 hours of 7080 computer time at each AMA. The IBM 1401 D041 report printing time for all of AFLC is 129 hours. Thus, the system used as a test problem during the experiment is a large one.

The D041 System is divided into several subsystems or modules. Each of these modules and the output products will be described briefly after the inputs to the system are listed.

(a) Inputs: The D041 receives input data from several other data systems and accepts manually inserted data from the item managers. A few of the types of input data are listed below:

(1) Stock Balance and Consumption Report lists usage data (the number of parts repaired or condemned, etc.) for each item for particular periods of time (called base periods) and the quantity of stock on hand as of a specific cutoff date.

(2) The Due-In-Asset Report lists quantities of stock items scheduled for delivery during a specified time period from outside sources such as under procurement contracts.

(3) Past Programs specifies the activity level (e.g., number of flying hours) during the base periods of the applications (e.g., weapon system) that the items are used on.

(4) War Readiness Material lists quantities of stock allocated for war reserve purposes.

(5) Project Programs lists planned activity levels of the application weapon systems during future time periods.

(6) Additive Assets, Requirements, and Factor Estimates are data that cannot be computed by the System from other inputs and must be entered by the item manager.

(7) Recovery and Condemnation Data lists data concerning number of items repaired or declared beyond repair during the base periods.

(b) Program Modules: The D041 System is composed of six separate modules that are run sequentially (with some feedback during recycling). A brief description of each follows:

(1) The data edit and file maintenance module is the input interface between the D041 System and other data systems and between the D041 System and the item manager. All data received are checked for errors according to established criteria. If inconsistencies are found, the data are printed for review by the item manager. The manager must supply (estimate in some cases) all elements of data that cannot be obtained from other sources, and he many override certain other elements of data. After the inconsistencies are eliminated, the master files are updated to incorporate the new data and delete obsolete data.

(2) The past program module develops a time-phased measure of the activity level during the most recent 30 months of the application for which the stock items are components. An example of a past program quantity is the number of hours flown by B-52s during a particular base period month. These quantities are developed from data supplied by HQ AFLC and certain other specified AMAs. The AMA and HQ AFLC data are merged and maintained in the D041 System master files.

(3) The factors module computes the applicable rates for various types of factors which individually are computed for each of several different types of usage data (e.g., base repair, depot condemnations, etc.) by dividing the usage quantities for a base period by the appropriate base period past program. These factors are then checked for compatibility and printed for review by the item manager. The item manager may override the computed factors by submitting new ones for a recycle through the initial modules of the system. The manager-entered factors will be used during the then current requirements computation, but will not be retained for use in future quarterly computations.

(4) The future program module is similar to the past program module except for the time period involved. The future program predicts the activity level of applications for the succeeding five years. These future program quantities are based on projections made by HQ Air Force.

(5) The requirements computation module computes the gross and net quantities of stock items necessary to support the application during the succeeding five years. The gross quantities are time-phased and are computed by multiplying the forecasted factors by the time-phased future program after suitable interpolation has been accomplished. Separate requirements are listed for each type of usage. The net requirements are produced by successively subtracting the the time-phased assets from the individual requirements according to the priority of the requirement. Thus, projected stock overages or shortages are determined. Dollar values of these projected overages and shortages are also computed.

(6) The Requirements Inventory Analysis Report (RIAR) module summarizes and regroups the information produced by the requirements computation module.

Other information helpful in deciding appropriate buy, scheduling, or contract termination actions is also produced.

(c) Output Products. The Requirements Inventory Analysis Report (RIAR) produced by the final module of the D041 System is the most important output of the system. However, there are approximately 50 other printed reports produced at various stages in the system that are used for validity checks, auditing, cross-referencing and various logistics management purposes. Many of the reports are similar, differing only in the format or sorting key.

The remainder of this report documents the building and use of the ADAM-based Requirements System. Section II reports on the data base analysis effort. Section III contains a description of the ADAM-based Requirements System. Section IV discusses the process of data base preparation and the characteristics of the ADAM System that the MITRE personnel implementing the ADAM-based Requirements System found useful or in need of improvement. Section V is written by the user of the ADAM-based Requirements System and contains comments concerning good and bad features with suggestions for improvements. The final section contains the conclusions that the authors feel can be derived from their experiences during the course of the experiment.

SECTION II

ANALYSIS OF THE DATA BASE

Each Air Material Area (AMA) has responsibility for a set of inventory items and maintains its own data base. When the AMAs' seven data bases are combined and the overlapping information removed, the resulting total is extremely large. Table I shows that, for the 67-1 data,* the total was approximately 164 million characters. It was not feasible, nor even desirable, to incorporate the entire data base into the ADAM-based experiment. Instead, a subset of the data base would be used. The composition of the subset was to be specified after the D041 data base was analyzed.

---

* 67-1 is an abbreviation for the first quarter of fiscal year 1967 and represents the D041 Computational cycle for that time.

Table I

Total 67-1 Data Base Volume

All AMA

File Name	No. of Physical Tape Records	No. of Characters Per File (in millions)	Percent of Total
Past Program	1378	3.9	2.38
Future Program	987	2.9	1.77
Item Past Program	8274	24.7	15.09
Application	2607	7.8	4.77
Index	680	2.0	1.22
Asset/Usage	31420	92.4	56.44
Technical Data	10181	30.0	18.33
TOTALS		163.7	100.00

The most important characteristics of the data base had to be isolated, extracted, and studied for the purpose of obtaining meaningful subsets. That is, a subset must be representative of the complete set and independent of other subsets. The data elements and formats of the master files were given in AFLCM 300-4 "Category I and IIR Consumption Items Requirements Computation System;" but actual file characteristics, e.g., tape formats or volume estimates, were not given. Neither the definition nor relative significance of the data elements were stated explicitly in that or any other available documentation.

To obtain specific answers, several special purpose data reduction routines were written. These routines performed three major functions; to print tapes in specific formats; to tally on various properties in files; and to correlate property values between two different files. These routines are briefly summarized below:

(1) Technical Data File Data Reduction:

The design of the tape file is based on a unit record; for items with data exceeding the fixed record length, more than one record is required. This routine recognizes an entity and tabulates the number of records and number of applications per item.

(2) Future Program Data Reduction:

This routine produces a tally of the number of applications and the frequencies of different values of Service Code, Cost Category, Record Identity, and Type Program.

(3) Asset/Usage Data Reduction:

This routine produces tallies and frequency distributions based on Master Stock Number (MSN), Actual Stock Number, and Record Type.

(4) General Purpose Data Reduction:

Based on any property in any file, frequency distributions of logical records per primary value and of values in secondary, tertiary,... fields are produced.

(5) Future Program Print:

The routine produces output in two parts. The first part contains the first 120 characters of all records, the second contains the last 80 of all records.

(6) Past Program Print:

The output of this routine occurs in three parts. The first section prints the first 100 characters, the second shows the next 70, and the third shows the last 90 characters.

(7) Application and Past Program Compare:

This routine determines which and how many of the application numbers on the Application file appear on the Past Program file.

(8) Application and Future Program Compare:

This routine functions similar to (7) using the Future Program file.

(9) Past and Future Programs Compare:

By considering the application number and Program Code, this routine tallies items common to the two files.

The information obtained by operating the named programs on sample data and 66-2 data* was instrumental in determining the criteria for extracting a subset of the data base. The final procedure consisted of finding all items related to the specified application(s) and is fully described in References 2 through 5.

The data reduction programs also provided vital information for determining the file design within ADAM. Some design considerations were the merging of the Tech Data and Asset Usage files; the separation of the Assets section from the Usage section in the Asset/Usage file; the reorientation from stock number to application number as the primary focus of the system. The final file organization is also described in References 2 through 5.

---

*66-2 represents the second quarter of fiscal year 1966.

SECTION III

ADAM-BASED REQUIREMENTS SYSTEM

GENERAL DESCRIPTION

Early in the design phase, the ADAM-based requirements computation accepted several functional constraints which served as a frame of reference for the general goals and design of the system. [6] One constraint was that both the data base structure and computational components remain essentially unchanged so that qualitative comparisons might be made. Another condition was that the computational functions be modular and so written that experimentation on individual modules would be feasible. To achieve this flexibility, all computations were implemented using FABLE, the ADAM query language.

One capability in the final system was not provided. Reference 6 outlines that the Item RIAR data would be made available for querying as a substitute for being able to perform the computations for all members of a subset. In concept, this additional file would have been generated without problem. However, an Item RIAR file would have had to be kept separate from all other files in the system so as not to invade those temporary files needed by the compute functions. Because the raw data needed special processing not allowed for in the normal preprocessing procedure, there were not enough resources to convert the data and generate the file.

The on-line, remote operations required explicit procedures. These procedures and/or functions which were needed to support the remote operation are presented later in this section.

DATA-BASE SUBSETS

Three different subsets of the 66-2 data were selected and produced as files within the ADAM context. Although the selection of the subset members was made arbitrarily, it was done with the aim of obtaining a representative sample of the application-stock number combinations in the data base. These three subsets are fully described. [2,3,4,5] Since the source data from which the subsets were derived changed during the project's life cycle, many subsets were started but never completed.

Seven distinct subsets of the 67-1 data were selected and specified by the AFLC Users Group. In three cases, a method for detecting items peculiar to a family of applications was given, but because of speed improvements and space savings attained by ADAM, total and complete subsets could be generated. The remaining four subsets were specified by particular values of the Federal Supply Classification (FSC) and the Material Management Code (MMC). These subsets are summarized below:

(1) B52

All items, both common and peculiar, were selected by means of the application tree algorithm described earlier. The particular names which were used to perform the subsetting operation were:
B52, B52A, B52B, B52C, B52D, B52E, B52F, B52G, B52H, B52T

(2) F4 and RF4

All common and peculiar items were selected using the same application tree program. The particular members of this family were:

F4, F4B, F4C, F4D, F4E, F4T, RF4C.

(3) C130, C135, KC135, C141

All common and peculiar items were extracted from the full data base by use of the application tree program. The family members of this subset originate from:

C130, C130A, C130B, C130C, C130D, C130E, C130H, C130T, C135, C135A, C135B, C135F, C135T, C141, C141A, C141T, KC135A, KC135B, EC135A, EC135C, EC135G, EC135H, EC135J, EC135K, EC135L, HC130E, HC130H, NC130B, NC130E, RC130A, RC135A, RC135B, RC135C, RC135D, RC135E. This subset was nicknamed CARGO.

(4) FSC 1270 and 1280 represent fire control items from WRAMA. Without use of the application tree algorithm, items and their immediate applications were selected when the FSC value was 1270 or 1280. This subset is named FSC12.

(5) FSC 5821, 5841, and 6615 encompass bomb navigation, communications and instruments from WRAMA and MAAMA. Those items and their immediate applications whose FSC value was 5821, 5841, or 6615 were selected. This subset is named FSC58.

(6) FSC 1650 and 1680 represent miscellaneous accessories from OCAMA and SAAMA. Items and their immediate applications were selected when the FSC value was 1650 or 1680.

This subset is known as FSC16.

(7) MMC

When the MMC was PD, PH, PJ, or PL, items and their immediate applications were selected without use of the application tree algorithm.

When all the above seven subsets were near completion, including the file maintenance function itemized later, some of the AFLC users requested a subset based on the F105 weapon system family. By specifying F105, F105B, F105D, F105F, F105T, and RF105D, a complete and independent subset was obtained by use of the application tree program.

Because internal storage space constituted one reason for working with data base subsets, the ultimate sizes are presented in Table II. The sizes of the 66-2 subsets are reproduced [2] so that comparisons with the 67-1 subsets may be made.

Table II

File Sizes and File Generation Times

66-2 DATA			
Subset	Input Volume (Physical Records)	Total Size (Arcs)*	Generation Time (Minutes)
F106	1831	952	433
10 Application	1830	897	376
26 Application	3702	1771	1065
	67-1 DATA		
MMC	1550	1264	203
FSC12	2954	1801	287
B52	4324	2506	390
F4	1473	836	138
F105	1212	870	142

* An arc on disk storage consists of 512 64-bit words.

Appendix III contains detailed information. In terms of number of objects (about 1200 items), the F106 subset from 66-2 is of comparable size to the F105 subset from 67-1 data. The B52 subset with 3324 objects may be contrasted with 2362 objects of the 26 Application subset.

PREPROCESSING

Each of eight AMA's contributed eight master tapes of 66-2 data to the total D041 data base. All tapes required multistep processing before the data base was ready for file generation in ADAM. Certain generalized routines were already available, and the rest of the needed programs had to be designed and written.

The overall procedure is shown in Figure 1, and Appendix VI gives a more detailed picture. The overall process consisted of copying all tapes received from the AMA's; sorting and merging all eight AMA versions of every master file; removing redundant records from the combined files; operating the two components of the subsetting algorithm; extracting the chosen members of the subset from each of the master files; and preprocessing each subsetted file in preparation for file generation. In the case of the TECH-DATA file, an extra step was needed to combine data from three different files.

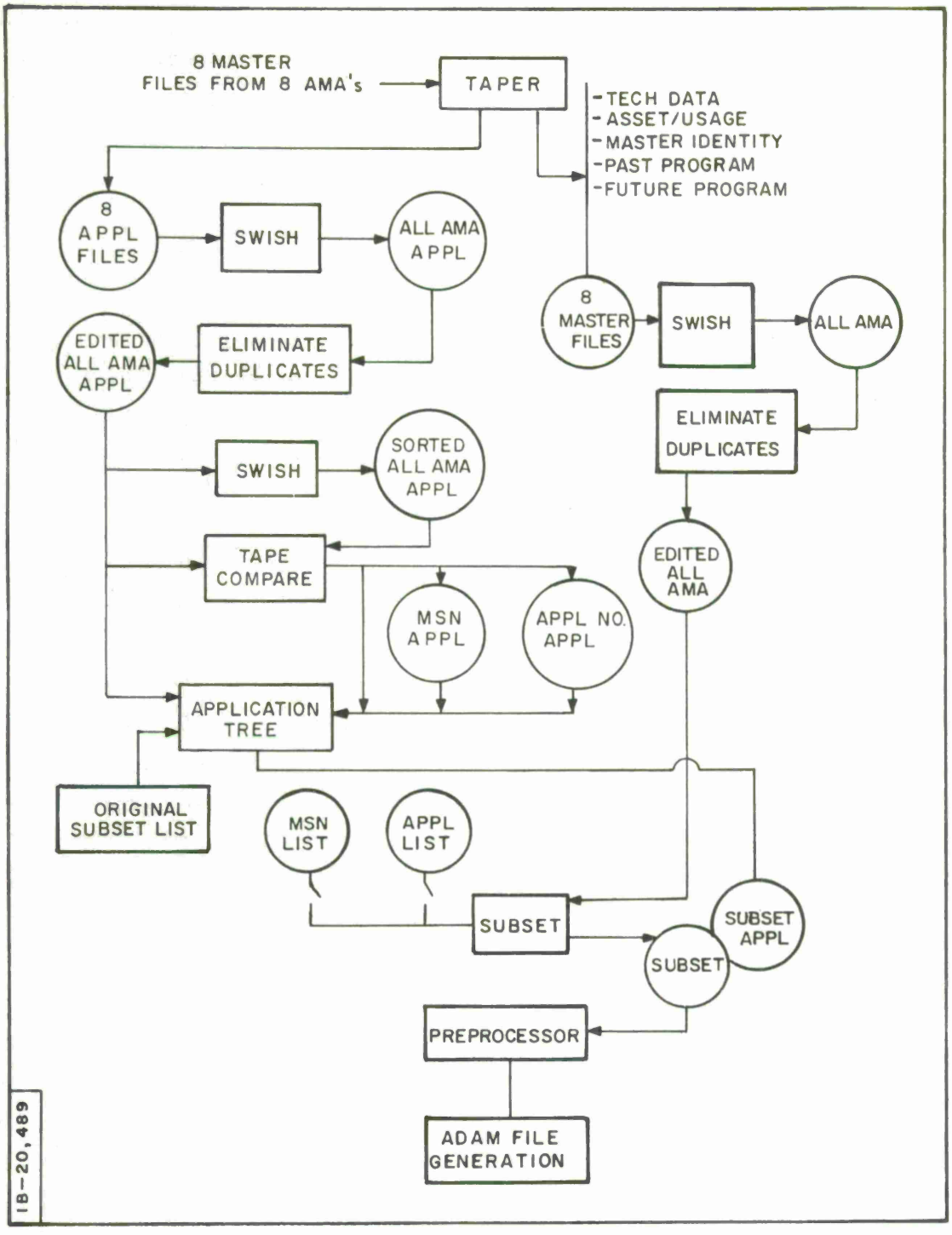


Figure 1. Processing of 66-2 Master Files

All processing steps may be categorized into those which were required to obtain a subset and which permitted the data to conform to the file generation language rules. Copying tapes was an obvious step. Both sorting and preprocessing were mandatory for a proper interface with ADAM. All other program functions were for the purpose of obtaining a subset.

After the preprocessing procedure was designed for the 66-2 data set, several unanticipated constraints were imposed: the subsetting algorithm was to be augmented; the master files changed substantially in format and content; and the number of different master files and their identities differed for this version of the data base. Originally, once the 66-2 data base was processed, the ADAM-based experiment might incorporate file updates and maintenance. When the decision was made not to include file maintenance, preprocessing of another total data base was mandatory. For all the above reasons, the preprocessing procedure needed several alterations.

Overall, the 67-1 data preparation is still much the same, but the program elements which comprise the total procedure are different. Tapes arriving from the AMA's were copied as before. An extra step was required when it was discovered that several logical files resided on the same physical reel and that these files were not separated in the conventional manner by tape marks. Then every master file was sorted and merged across the AMA's. Special programs were required to eliminate records which were judged redundant after the first file was generated. Once the subset had been determined, the appropriate members had to be extracted from each of the all-AMA master files. These subsets were then preprocessed for file generation in ADAM.

Although the subsetting algorithm [2] was still the recommended method for obtaining a complete and independent subset, AFLC requested other ways of extracting a data base subset. The users were interested in basing a subset on certain values of particular properties in the files. In particular, the properties were Material Management Code, Budget Code, and Federal Supply Classification. In view of this, a program was built which could accept several values of any property to select a subset.

FILE GENERATION

The Initial File Generation Language (IFGL) is one of the user languages provided by the ADAM system. By use of IFGL both the eventual structure of the file and the raw data characteristics may be described. The whole file description written in IFGL was considered a message to be translated and processed by ADAM. One IFGL message per D041 file component was written. The final file components were Past Program, Future Program, Asset/Usage, Technical Data, and Applications. The structure of these files is described in Reference 2, as are the IFGL messages which were used to generate the files for both the 66-2 and 67-1 versions. The actual implementation is described later in Section IV under File Generation. The properties which comprise each file are listed in Appendix IV.

COMPUTATIONS

For purposes of computing future requirements of inventory items, a base period is defined, usually as a 24-month period ending three months prior to the asset cutoff date. Past program data and certain usage data are collected for the base period on a master-stock-number basis. These 24-month accumulations are used to calculate factors. The factors are used in the requirements calculation, in conjunction with future program data and current asset data, to estimate future requirements.

Certain of the factors may optionally be entered by the item manager rather than calculated. There are four basic steps in the computation of factors:

(1) The item manager must enter forecast values for four of the factors.

(2) The manager-entered, forecast factors are used to calculate forecast values for Base Repairable Generations, Base NRTS (not repairable this station), and Base Condemnations from the base-period values of these three quantities.

(3) The system uses the base-period and adjusted base-period values listed in (2) plus base-period values of the Past Program, Depot Condemnations, and Depot Repairable Generations to calculate the remaining factors. Four of the factors are rates of failure or repair per unit of program; and the remaining factors are percentages used to calculate certain future requirements and assets.

(4) For each master stock number, the results of the factor computations are printed.

Material requirements on a quarterly basis for each master stock number are computed for a nominal five-year period. In general, at the time the requirements computation is begun, a current value and three forecast values must be available for each type of factor. Essentially, an average of adjacent factors is used to compute a more accurate requirement for the year bounded by these two factors.

Additional inputs to the requirements computation are:

(1) future program data (MRS, OFM, IRAN, Military Assistance Program) which are converted to Item Future Program using all final applications of the item;

(2) technical data consisting of repair cycle days, negotiated base and depot stock levels, and unit prices;

(3) dated quarterly values for those assets which are not computed by using factors; and

(4) additive requirements entered by the item manager.

The first task of the gross and net compute function is to calculate the total gross Air Force requirement, which basically is composed of three elements:

(1) materiel to satisfy various stock-level needs;
(2) materiel to replace condemned materiel; and
(3) additive and war reserve requirements.

MAP requirements are determined separately from those of the Air Force, since Air Force requirements must be fulfilled before satisfying those of MAP.

Five classes of assets are aggregated and applied separately as offsets against the requirements. Air Force and MAP over and short positions are computed at each level by applying a class of assets against the requirements. The asset classes are applied in the following order: serviceable assets; base repairs; TOC; depot repairs; and on-order assets.

Following computation of the fifth Air Force/MAP over and short positions, various non-time-phased quantities are calculated. Buy, termination, and excess quantities and dollar values are determined using results from the earlier calculations. Various worldwide, depot, and distribution levels are then calculated.

The requirements computation also produces a file containing basic data which in turn is used to produce the Requirements Inventory Analysis Reports (RIAR).

Complete descriptions of all computations comprising the ADAM-based system are given in References 7, 8, and 9.

REMOTE OPERATION

In order to test the performance of ADAM through the reactions of an actual user, input-output devices were installed at AFLC Headquarters in Dayton, Ohio, which would provide a direct connection to the 7030 Computer at MITRE through regular telephone lines. This provided AFLC with the capability of direct, on-line querying of any one of their subsets residing in ADAM. The Remote Operation consisted of the following segments:

(1) installation and checkout of remote equipment;

(2) training of the users in the fundamentals of ADAM, FABLE, remote equipment procedure and minor maintenance; and

(3) administering and monitoring of the on-line remote computer time.

The Remote Equipment was specified to consist of a modified Stromberg Carlson 3070 high speed printer and a model 35 ASR even parity teletype with data phone.

The equipment was installed by 1 April 1966; the SC 3070 printer was provided and installed by MITRE, while the installation of the teletype was performed by Ohio Bell Telephone. However, the teletype provided by Ohio Bell Telephone was a converted TWX machine with odd parity and was not compatible with the 7030 adapter. This delayed the successful operation of the remote station by four days and required a modification of the adapter. The requested model was finally installed 9 May 1966 and the necessary remodification of the adapter was made.

A short course on the maintenance of the SC 3070 printer and the 35 ASR teletype was given to project personnel who then trained the users at AFLC. The instruction of AFLC personnel in the principles of ADAM and FABLE took place over a two-week period at Dayton. Several classes were held in which FABLE was discussed in the form of lectures. It was then recommended that prospective users of the system should contact the visiting MITRE personnel on an individual basis with their particular questions and problems. It should be noted here that, during the second week of the instructions, the Remote Station was fully operational and was online one hour daily.

All remote runs were monitored and the set-up controlled by MITRE at MITRE Command Post B. "Control" means the preparation of the card deck with the associated selection of the subset.

To perform effective monitoring, at least two SC 3070 printers and two INVAC typewriters had to be present in the configuration. One printer was used to monitor all incoming messages from all devices; the second printer monitored the messages going to the remote printer. The need for two typewriters is due to the fact that one should be used only to receive control messages such as AUTO RESTART BEGUN. This message could be lost if it were sent

to that typewriter while it was in input mode. While the devices listed above establish the minimum configuration, the addition of further units usually increased the effectiveness of monitoring.

In order to test the operational readiness of the remote devices, the Equipment Checkout program (ECO) was operated before the ADAM-based system was loaded. This routine and a full description of its capabilities are reported in Reference 10.

SECTION IV

IMPLEMENTING THE ADAM-BASED REQUIREMENTS SYSTEM

The implementation of the ADAM/AFLC experiment encompassed several large areas of endeavor. The major segments are identified and discussed. In particular, some of the procedural or mechanical difficulties, whether related to data preparation or on-line operation, are reported. ADAM characteristics which were not necessarily suited to the application are identified. In general, the section is devoted to describing the total process of implementation.

Preprocessing of AFLC data both to produce the appropriate data base and to force compatibility with ADAM represented a large portion of the total project effort, e.g., to program and checkout all special purpose programs required 17 man-months. The development of computer programs and processing procedure is described. This subsection is distinct from any critique on ADAM characteristics or concepts represented by such a system.

The general structure of ADAM allowed certain tasks to be performed extremely well. The concept of independent tasks, the existence of routines in files, the independence of data bases from the system - all these elements were very useful for such tasks as generalized retrieval. Some of the generalized capabilities provided by ADAM were either not utilized or fully taxed by the construction of the ADAM-based D041. For example, ADAM provides random access mechanisms, whereas efficient D041 operation required serial processing of large amounts of data; or routines could be written and stored in files whereas the implementation was actually accomplished in the query language, FABLE. The particular capabilities used in implementing the experiment are delineated in the subsections on file generation and maintenance, the computations, user aids, and special routines.

DEVELOPMENT OF GENERALIZED PROGRAMS

The 66-2 processing procedure required multiple passes over the same data in order to obtain the all-AMA files. Since these multiple passes were conceptually unnecessary, a generalized sort/merge program could be designed and contain an input and/or output processor which could perform such functions as removing duplicate information and padding out variable length records to constant size. The benefit would occur not only in decreased 7030 time but also in turnaround and thus elapsed time.

It was also concluded that SWISH, a generalized sort program*, was not ideally suited for sorting AFLC data in bulk. Further, separate merge capability had been used without a sort. With the goals of saving 7030 and elapsed time, enlarging the capacity of a logical file, and providing a separate merge function, a new sort/merge program system was designed and implemented. The sort segment is known as DISKSORT** and the merge segment as MERGE. [11]

For the 66-2 data, SWISH was used to sort and merge the data from all the AMA's. Because of certain volume limitations, all the items could not be combined into a larger logical file. DISKSORT was used to process the 67-1 data. The design and implementation of the DISKSORT/MERGE and associated checkout programs required about 11 man-months. The results were rewarding. Some timing summaries are compared below for approximately equal volume per file type. Each quantity is the sum of all the runs which produced the final output, thereby reflecting the need for segmenting.

---

* STREAM System Manual, SR-114.

** DISKSORT will be described in a forthcoming report by J. B. Glore

	DISKSORT (MIN)	SWISH (MIN)
Past Program	6	55
Future Program	6	79
Item Past Program	23	258
Application	9	75
Index	2	Not applicable
Asset/Usage	30	628
Technical Data	23	379

PREPROCESSING PROCEDURES

Each of seven AMA's sent four sets of master tapes to comprise the total 67-1 Version of the D041 data base. The four master files, as identified by the AFLC designation were: C5A (multiple logical files containing stock numbers, application numbers, and part numbers); C5E (a single logical file, Asset/Usage); F1B (a single logical file, Technical Data); and J3A (multiple logical files of past program, future program, and item past program). B8A and C6A were sent by each AMA but were not used in the generation of the ADAM-based files.

The single file tapes, C5E and F1B, were copied by TAPER, a generalized tape handling program,* on the 7030 whenever possible; otherwise, the IBM 1410 was used when the 7030 detected too many parity errors. Since the various logical files on the C5A and J3A needed breaking out, the program SPLIT** was written to perform this function and to accomplish the duplication in the same run. At this point, 49 (seven master files from each of seven AMA's) logical reels of tape existed.

Each file needed to be sorted and merged across the AMA's. As indicated previously, the duplicate records were to be removed in the output processor of the DISKSORT and/or MERGE. Table III shows the required sort keys.

It was expected that there would be a high frequency of redundancies in the past and future program and application files and that the occurrence of duplication on the remaining files would be possible but far less probable. The expectations were realized when the past and future program and application files were each combined across the AMA's to fit a single reel. The index files portion was also reduced to a single reel. Although the remaining files each required multiple reels, each could be treated as a single logical file.

Duplication of data still existed even after the generation of the first set of ADAM files. In the case of Past Program, outdated information had to be removed. For the Future Program, some records contained a headquarters code which had to be ignored in order to remove duplicates. Thus, the two files were processed by the programs PURF and PURP. At that point, there existed all-AMA files for these distinct master files: Past Program, Future Program, Application, Index, Item Past Program, Technical Data, and Asset/Usage.

---

* 7030 Facility Manual, SR-76, pp. 315-322.1.

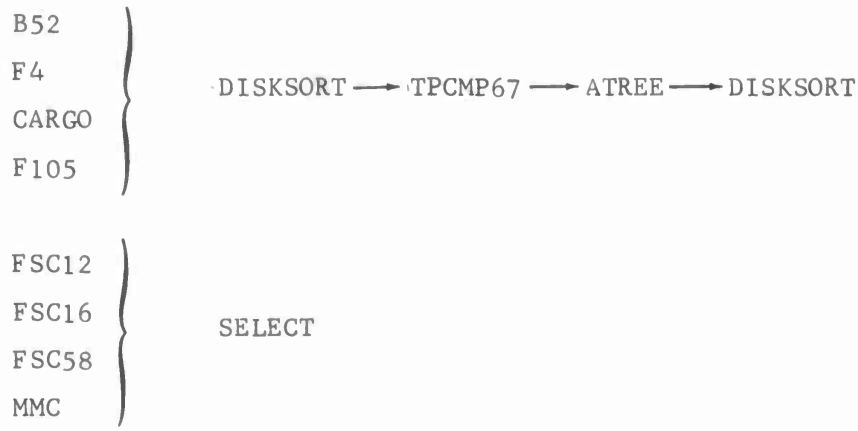
** All preprocessing programs are described in Reference 10. The specific program design, coding, and checkout required about 20 man-months.

Table III

Summary of Disksort Use

File Name	Past Program	Future Program	Item Past Program	Application	Index	Asset/ Usage	Tech Data
Item Length or Range (Characters)	260	210	230	60	70	110-210	470-590
Blocking Factor	11	14	13	50	42	14	5
Number of Keys	3	3	2	2	2	2	1
Name (Primary)	Appl.No.	Appl.No.	MSN	Appl.No.	Current SN	MSN	MSN
Positions	4-18	4-18	10-24	10-24	25-39	10-24	10-24
Name	Type	Type	Appl.No.	MSN	MSN	Rcd.Type	
Positions	20	20	25-39	26-40	10-24	41	
Name	Serv Code	Serv Code					
Positions	19	19					
Volume (No. of Physical Records.)							
Initial	1551	1620	9732	2734	1085	32,249	10,912
Final	1378	987	8274	2607	680	31,420	10,181
No. of Segments	1	1	1	1	1	5	3

The method for obtaining complete, independent, and meaningful subsets was described in Reference 2. The programs which perform the subsetting algorithm are TPCMP67 and ATREE. Modifications to these programs which were needed for the 67-1 data base consisted of input/output format handling changes. The augmentation of the subsetting capability as required by the AFLC subset specifications was implemented in a program named SELECT. Each of the seven subsets defined previously traveled one of two routes through the set of programs.



Either route yielded two products: a subset of the application file and a key tape for extracting the subsets from the other master files. Each of the all-AMA files, except the application file, was passed against a key tape by the program SUBSET. Thus, the production of a single subset required seven separate SUBSET operations using a key tape. It had been considered that this particular function of extracting the actual subset from the master files might be accomplished within ADAM. The method would have consisted of generating the whole of each file separately, saving each file on tape; generating the required object roll from the key tape and, at file generation time, passing the total file against the object roll thereby doing the selection. These schemes were not feasible because of memory size limitations.

As can be seen in Appendix VI, the TECHDATA file was the only one which was immediately acceptable by ADAM. The rest of the files required further processing by the PREP program, and in the case of the application file, by program ANNEX. The major functions performed by PREP are summarized by master files below.

	FUTPROG	PASTPROG	ASSETUSG	TECHDATA	APPL	INDEX
Repeating group terminator insertion	x	x	x		x	x
Shifting fields of data	x	x	x			
Removal of all zero fields			x			
Removal of leading and terminal zero fields	x	x				
Aggregation of data across actual stock numbers			x			

The functions which ANNEX performed consisted of recovering time-phased data from the Item Past Program file, correlating this data with the Past Program file, and recording the results in abbreviated form on the Application file. The ANNEX operations prepared the major data deviation of the ADAM-based file structure from that of D041.

Appendix VI shows the overall procedure. Note that the portion to the left of the dotted line represents those steps which need be performed only once for each set of data received from the AMA's. The right-hand portion must be done for each subset.

FILE GENERATION

The sets of files for the subsets of AFLC 66-2 data were large, and it was anticipated that the 67-1 subsets would be even larger. Storage space was one consideration. File generation time could also be prohibitive. For the assumed confinement of subset size to 4000 to 5000 stock numbers, the space problem was solved when the maximum allowable disk space was doubled to 4096 arcs. By way of file design, space was saved in two ways. Stock number and application number relationships in the 66-2 data base were given in both the TECHDATA and APPL files; for the 67-1 data only the APPL file contained that information. The second way of conserving space consisted of aggregating the data in the ASSETUSG file across the actual (current) stock numbers.

Appendix III gives explicit data about file sizes within the ADAM environment. Add to each total about 600 arcs which is the space required for the ADAM system itself under normal circumstances. In performing the D041 compute functions, several temporary files are required. These computations were performed on three items in the F105 subset, and the space used was measured: 31 arcs for data and 14 arcs for rolls. The 45-arc total is simply a sample of the additional storage space needed.

A major modification to the file generation process within ADAM was implemented between the receipts of 66-2 and 67-1 data. Not only did this change realize a time savings ratio of up to five to one but also, since the data volume to file generation time relationship was linear, made the estimation of required computer time possible. The file generation times are given in Table II.

For the major reason of providing economical cross-file referencing, certain properties used the object roll of other files during file generation. A prescribed order was necessary for generating the six files which comprise a subset. FUTPROG, PASTPROG, and ASSETUSG files could theoretically be generated in any order, but were created in that order by convention. The TECHDATA file was generated next; a property was created that used the object roll of the ASSETUSG file. The APPL file was next and contained properties which used the object rolls of all preceding files. The INDEX file, containing a property which used the object roll of the APPL file, was generated last. Below, OR means "object roll."

File Name	Property Name	Roll Usage
FUTPROG, PASTPROG, ASSETUSG	Not Applicable	
TECHDATA	MSNAU	OR of ASSETUSG
APPL	MSNAU	OR of ASSETUSG
	MSNTD	OR of TECHDATA
	APPLNO	OR of APPL
	APNOPP	OR of PASTPROG
	APNOFP	OR of FUTPROG
INDEX	MSNAP	OR of APPL
	MSNIN	OR of INDEX

An additional file was generated at the request of AFLC users who required inventory and usage information by actual stock number. This ASUS file was almost identical to the ASSETUSG file. The property names were the same but there were some structural differences. The time-phased data had not been

summed across actual stock numbers. Other files did not reference its object roll. The actual stock number was used as the object name, and the master stock number (MSNAP) was a main level property cross-referenced to the APPL file. If such a file were needed permanently in the data base, it would be merged with the information contained in the INDEX file.

Whenever feasible, the most current version of the Production ADAM System (PAS) was used for file generation. The generated files themselves became independent from the generating system as soon as they were saved on tape. The reason for using the most current PAS, then, was solely to take advantage of all improvements on and corrections to ADAM. The components of any current PAS were the latest PAS tape, a standard card deck, and insertions to this deck composed of modifications. The one special ingredient of the operational setup consisted of a conversion routine, AFLCON, which converted zone bits of a six-bit byte into an algebraic sign for integers and floating point numbers.

For any one subset, each file type had to be generated as a separate job in the prescribed order. Procedures exist and were used successfully to generate multiple files in sequence. The advantages in multiple generations were the computer time saved in not having to restore already generated files,and the turnaround time (e.g., 24 hours) by submitting only a single job deck to the IBM 7030 queue. The disadvantage of a multiple generation was the complexity of the deck setup for mounting new raw data tapes and new scratch tapes for saving.

Except for the FUTPROG file,which was the first file into a data base, the generation of each file type required that all the predecessors to be restored onto disk. For example, the tape containing FUTPROG and PASTPROG was restored before the file ASSETUSG

was generated. At the completion of a file generation, all files up to that point were saved twice on two separate tapes. In the case of the B52 and CARGO subsets, the whole set of files would not physically fit on a single reel of tape. This fact required that the APPL file be saved separately. The procedure sheets for the F4, F105, B52, MMC, and FSC12 subsets are reproduced in Appendix V.

Once all six files comprising a subset have been generated, there are still further operations to be performed so that the files will be as current and clean as possible. All of these operations can be performed with FABLE as the file maintenance language. Certain special routines were written to speed up processing. The particular FABLE statements are enumerated and their function described below.

(1) Messages which add values to the roll PROP:

(a) FOR ASSETUSG. ADD VALUES M,N,O,P,Q,R,S,T,U, V,W,L TO ADDAS NAME.

(b) FOR FUTPROG. ADD VALUES 1,2,3,4,6,7,9 TO FUTOFM PC.

(c) FOR TECHDATA. ADD VALUES L, LM TO MRLC.

(2) Messages which set certain numeric properties to constant values:

(d) FOR TECHDATA. SLBOFM LQ Ø AND (ERRC EQ T CHANGE SLBOFM TO 3Ø OR CHANGE SLBOFM TO 8).

(e) FOR TECHDATA. SLDOFM LQ Ø AND (ERRC EQ T CHANGE SLDOFM TO 3Ø OR CHANGE SLDOFM TO 15).

(f) FOR TECHDATA. SLAA LQ Ø AND (ERRC EQ T CHANGE SLAA TO 6Ø OR

CHANGE SLAA TO 3Ø).

(g) FOR TECHDATA. SLOH LQ Ø CHANGE SLOH TO 3Ø.

(h) FOR TECHDATA. CHANGE ACUTYR TO 66, ACUTMO TO 9.

(3) Messages which delete certain zero-valued repetitions:

(i) FOR ASSETUSG. IF UQUANT EQ Ø DELETE REPETITION UQ.

(j) FOR APPL. IF FYR EQ Ø DELETE REPETITION ITEMPP.

(4) A set of messages which can eliminate redundant repetitions in the APPL file and provides a count of redundancies per object.

(k) LET MATCH MEAN (APPLNO EQ A APPLNO AND PSC EQ A PSC AND PPC EQ A PPC AND QPA EQ A QPA AND PASTPC EQ A PASTPC AND FUTPC EQ A FUTPC AND ERRC EQ A ERRC) FOR ALL USING RESCAN.

(l) LET A MEAN (APPL (MS)) FOR ALL.

(m) FOR APPL. NOT ELSE, SAVE 'MS' = OBJECT NAME, APPLNO, PSC, PPC, QPA, PASTPC, FUTPC, ERRC, 'X' OF APPLNOG = Ø. NAME TEST.

(n) FOR APPL ALTER TEST. ADD OBJECT MS = OBJECT NAME, APPLNOG (APPLNO= APPLNO, PSC = PSC, PPC = PPC, QPA = QPA, PASTPC = PASTPC, FUTPC = FUTPC, ERRC = ERRC, X = Ø).

(o) FOR TEST. FOR APPL (MS) APPLNOG. ANY (MATCH CHANGE X TO X + 1).

(p) FOR TEST ALTER APPL (MS). FOR APPLNOG. ANY (MATCH AND X GR 1 DELETE REPETITION APPLNOG).

(q) FOR TEST. TALLY FOR X, EQ Ø$1$5. PRINT FORMAT TALLY TALLY.

(5) Message which transfers the price from TECHDATA TO ASSETUSG for each object:

(r) FOR TECHDATA ALTER ASSETUSG (MSNAU). CHANGE PRICE TO PRICE.

For extremely large files, this message can be very time-consuming. A special ADAM routine was written and may now be triggered by the message: DO JAN ( ).

(6) Messages which operate a special ADAM routine and recover the space formerly occupied by now-NULL data:

(s) DO REPCO (ASSETUSG, Ø).

(t) DO REPCO (APPL, Ø).

(7) Messages which serve as checks that the files were generated and cleaned before making the subset available to the remote operation:

(u) LASTNOBJ filename no. of objects

(v) FOR filename. ELSE UNTIL 2, PRINT ALL.

(w) DO MELD ( ).

(x) DO POINTS ( ).

} These operate routines which output information about file sizes.

COMPUTATIONS

A summary of the D041 compute function appeared in Section II, and the products of the ADAM-based implementation are documented in References 12 and 13. A representative sample of the capabilities of the ADAM-based system is reported in Reference 14.

The relevant elements of a discussion on such an implementation process have to do with the particular language and the eventual execution time on the central processing unit. One capability of ADAM allowed the user to program in a lower level language those functions not suited to a message language. For this project, the decision was made to forego that capability in favor of FABLE whenever possible. Due to the trial and error method required to emulate the D041 function, the flexibility of the FABLE language reduced that impact of uncertainty. Although specific dimensions are not isolated in this section, some commentary on language and processing time is pertinent.

Although the FABLE language is extremely powerful, at least sufficiently powerful to perform the D041 computation, there are several drawbacks in its use. By design, it is a language which is oriented to a single task, is independent of other messages, and is translated each time. These characteristics, for this application, were the source of some difficulties, and because of them, many sections of the computation had to be written in an unconventional and sometimes simplified form. To implement the computation, the individual queries had to be strung together to form a program. This program was compiled every time and contained only internal query looping.

If used properly, FABLE can perform almost all programming tasks a lower level language can perform. Subroutines may be created using string substitutions with parameters and nesting. Files may be indexed, specified objects may be addressed, and by use of cross-file referencing, tabular matching may be performed. Essentially, FABLE lacks the ability for intertask branching and off-line assembly to make it a really useful programming tool.

Because the FABLE language can use nearly all of the features of ADAM, only two independent DAMSEL routines were needed to perform specific tasks (i.e., a square root function and a simplified query-calling program) for this application.

FABLE, in its entirety, is an extremely difficult language to learn, and its use can confuse even the most experienced user. Its use as a simple retrieval language can be hampered by this complex structure. The nonfamiliar ADAM user has much difficulty in grasping the use of this language. To make it a more understandable retrieval language, it needs only to be simplified in both verbiage and syntax.

Execution time of the computations showed that translation was the most time-consuming of all operations. If a query took one minute to complete, usually 55 to 58 seconds were for translation and 2 to 5 seconds for processing. The actual processor time for most queries was small compared to the translation and control procedure time. In running the computations on less than 4 or 5 objects, the overhead time consumption by the translator was high; but as the number of objects increased, the computation time was used up more for processing than translation. This implies that the most efficient way to run the system would be for many objects at one time. However, the total time becomes prohibitive for very many objects.

Although in concept any number of stock numbers could be used for a computation, the other limitation has to do with hardware and not software. If the 7030 had unlimited core and disk space, then the computation could be performed for any number of stock numbers. In reality, however, the ultimate limitation was based on computer time.

Individual query execution times are given in Reference 12. However, some timing figures for the Gross and Net Computations section are summarized as follows:

Execution Time (in minutes)			
No. of Stock Numbers	Translator	Processor	Output
2	21.75	4 33	2.50
32	21.75	44.25	9.16

While the operation of the set of computations was time-consuming, relatively little effort was required to code and checkout the ADAM-based programs -- about five man-months. These FABLE routines replicated the factors, requirements, and RIAR computations of D041, which segment consists of approximately 15,000 symbolic language (AUTOCODER) instructions.

No specific experiments to speed up execution time by using DAMSEL were conducted. On page 42, the routine JAN was mentioned. This routine performed in five minutes a function which required 40 minutes. It took advantage of the serial structure of the file.

ADAM provides a standard format suitable for all output devices. Should this standard format be unsuitable for a particular application, ADAM also provides an output formatting language. This language and its associated macros proved flexible and effective. PPCONV was the only lower level language program required to produce all desired output formats.[16] Although the particular device for which the format was being designed was irrelevant, most of the D041 formats were geared to the SC3070 since that was part of the remote configuration.

REMOTE OPERATIONS

The modifications to any software for operation with a remote high speed printer and teletype were relatively minor. Some changes to the Master Control Program (MCP)* were needed to adapt the system to the teletype.** The ADAM system required only that a new device be defined. These changes were all ready by 1 April 1966.

Related to mandatory changes to software were changes to install some desirable capabilities. The most important addition was made to allow monitoring of the remote operations on equipment in Command Post B. The slaving of like devices, e.g., the remote SC3070 to a near SC3070, was trivial due to the design of the relevant portions of ADAM. In contrast, the slaving of a nonteletype to the remote teletype presented some software problems. When these were solved, another near SC3070 acted as monitor for the remote teletype. However, in no way were messages from Dayton intercepted and edited before they were accepted into the system. The monitoring capability had no control functions until after messages were entered. At that, the only function the monitoring personnel could perform on a message from the remote station would be cancellation.

Another user aid consisted of message numbering, that is, the assignment of an identification number to a query. If a message were rejected by the system, its number enabled the user to relate the diagnostic to the transgressing input query.

The second major aspect of remote operation concerns the equipment. The configuration has been described earlier and was well suited to the goals of the experiment. Once the initial installation and checkout problems were solved, the remote equipment performed extremely well. The successful operation of the

* MCP is the operating system for the IBM 7030.

** See Reference 18.

ECO program increased the probability of reliable equipment operation.

The remote operation encountered problems with near equipment, rather than with the remote equipment. Each problem occurred only infrequently; but in aggregation, the problems were significant. Twice there were no display consoles (and thus no INVAC typewriters); many times there have been only two out of four complete units available. The SC3070's tended to produce illegible output. The typewriters were subject to character misinterpretation on input and produced occasionally scrambled or garbled printed output.

Under certain conditions, a vital communication link between AFLC users in Dayton and project personnel at SDL was missing. If the ADAM system was lost through either program or machine failure, no relatively fast way of communication existed. Voice connection is initiated by the remote station and not the command post, so no legal way existed for the command post to communicate with the remote station. Also, it was apparent that to keep the voice line open through the entire run would be unnecessary and at times ineffectual since communicants normally did not remain at the phone at all times. A great help would have been some capability such as a teletype hardware feature similar to the "HERE IS" key which, if depressed at the command post, would send a set of characters to the remote teletype instructing them to resume voice communication. This "alarm button" would have been very useful at many times during the remote running period.

After the initial two weeks of remote operation, the AFLC job was run as a foreground job allowing short background jobs to be run at the same time. Only one procedural change was made to release

tape units for use by the background jobs. One unit of measure consists of the number of background jobs operated; these figures are available. The total job time quantities are not available. In general, there was a decrease in the number of background jobs executed as the remote operation grew older. During May and June, the daily average was two, while during July the average was one per day. One reason was that the user's mechanical familiarity with the teletype increased, coupled with the fact that MCP originally did not buffer that type of input. The second reason was that the user's increasing knowledge of FABLE allowed for entering more sophisitcated queries faster.

USER AIDS

Only certain capabilities and deficiencies are mentioned in this report. The relative ease of normal on-line and remote operation testifies to the fact that the overall ADAM system has a wealth of user aids. The items mentioned here pertain to the most difficult mode of operation--debugging.

Query numbering was one debugging aid. But its vital necessity arose from the lack of meaningful error messages. The present system has relatively few error messages, and these are so general that the only aid they give the user is the fact that there is an error somewhere in his message. The error returns provide limited help to the experienced ADAM user; but in the case of AFLC personnel who are relatively unfamiliar with the ADAM mechanisms, the error messages provide, in many cases, no real help. What is needed is more error messages which are more meaningful and lack ambiguity.

The utility messages are recognized by the "$" which is always the first character, and internally are characterized by the fact that they bypass the translator. Certain of these messages have proven extremely useful for debugging. The most frequently used ones and the purpose of their use are listed below. The precise formats are not necessarily shown.

$EXPAND $CONTRACT	Erases and rewrites fixed routines in order to get more core space.
$TIME	Returns time of day and, since it is a bypass message, assures the user that the system is still running.
$WHICH	Returns the message number of the message being processed.
$TELL	Enables one device to transmit a message to another device.
$RECOVER $END	Ends the current task and initiates an automatic restart or starts a new task.
$UNLOAD $MOUNT $SYSTEM $RESYSTEM	These are various functions which are used when new subsets are desired or a restart tape must be saved.

DOCUMENTATION

For the effective implementation of this experiment, it was mandatory that project personnel become both problem-oriented and indoctrinated in the workings of ADAM. The textbook terminology relevant to inventory control and management was readily available. However, like any organization, AFLC employed a hybrid set of terms and phrases which differed from normal usage (e.g., slippage, stratification, extrapolation). Had a document been available to

explain these terms fully, ambiguities would have been minimized and the orientation period shorter.

There was the major task of learning how D041 operated and why D041 performed its various subfunctions. It was important that the basic operating philosophy behind the theoretical principles of D041 be known since the experiment was meant to duplicate D041's intent, not the whole computer system. Yet these principles needed to be stated at a level detailed enough for design purposes (e.g., not in terms of Department of Defense directives). At the other extreme, there existed very detailed program descriptions and system operating instructions. It was even necessary to read machine (AUTOCODER) listings to obtain a working knowledge of D041.

At the time of this writing, documentation on ADAM exists which is user oriented. [15] During the period when the experiment was being implemented, such literature was not available. What did exist were internal reports directed at programmers constructing the ADAM system itself. These reports, then, were excessively detailed for the user. IFGL was described in a manual which came closest to being a user's guide. In contrast, FABLE was not well described except in terms of syntax diagrams. FABLE is a complex and powerful language, and it was one of the goals of the project to write everything in FABLE for flexibility. The power of FABLE could not always be demonstrated using spontaneous queries. Although some project personnel became well versed via trial and error techniques, without adequate guides the imparting of this knowledge to users not co-located was difficult.

SPECIAL ROUTINES

During the implementation phase of the project, it was necessary to write several routines at a language level lower than FABLE or IFGL.

AFLCON was written for use during file generation. Its function was to convert the algebraic sign of integers and floating point numbers from a sign overpunch in the units position to a sign byte. A detailed description is found in Reference 16.

PPCONV was written for use in output formatting. Purely for aesthetic reasons, it allows proper handling of repetitions. [16]

APOWER is an exponentiation routine which was converted to an ADAM environment. The FABLE syntax was designed to handle exponentiation, but the capability was not ready. [16]

DEWALL was written to facilitate entering the computational queries and thus prevent mechanical and procedural mishaps. [12]

SECTION V

USING THE ADAM-BASED REQUIREMENTS SYSTEM

Use of the ADAM-based requirements system was accomplished by means of the remote operations described in previous sections. The User Reaction part of this section was written and submitted for this final report by the AFLC Users Group and is reproduced intact. The Usage Estimates part gives some average estimates of ADAM-based requirements system usage.

USER REACTIONS

Execution of the Experiment

Plans were developed for the installation of a remote station at Hq AFLC. Initially, the plans called for the training of all committee members in the use of the remote equipment, the FABLE language, and the ADAM system so that each person could write his own queries and make interrogations of the system,using the remote equipment. This portion of the test was to determine how mission personnel would fare as their own programmers and operators. Although a number of attempts were made by most members of the committee at writing their own queries, they were successful at only the simplest queries, and then quite often made mistakes of punctuation or in typing. In nearly all cases, the mission personnel were inclined to turn their more complex queries over to a committee member who was a programmer and had mastered the FABLE language and remote equipment to a greater degree.

After several attempts to encourage other members of the committee to prepare their own queries met with failure, plans were changed to correspond to the reality of the situation. That is,

some mission personnel, as a matter of principle, did not believe that it should be their function to learn a programming language, and others who would try did not have the time and/or patience to master the intricacies of the language and equipment, and became discouraged after repeated failure in attempting to use the English-like syntax query language.

The new plans called for the other committee members to feed their queries to the programmer-members. The queries were written in either straight English narrative or preferably in English narrative with all references to data in terms of the abbreviations used in the ADAM system. This approach worked reasonably well, but occasionally a backlog of work occurred, sometimes from the sheer volume but often because the programmer-members were stumped as to how to state a query. Quite often we were obliged to call upon our MITRE associates to "bail us out." At times, they would have to tell us it would be very difficult to write or too time-consuming to process. Many of the queries which the mission personnel asked required data not available in the ADAM D041 files. Both of the latter situations caused a reduction in the number of active users and in the number of queries submitted.

Other than the above-mentioned difficulties, the experiment progressed satisfactorily with the queries submitted by the committee members being converted to the FABLE language, keypunched on paper tape and processed by one or two committee members. The equipment at the AFLC remote site, which consisted of a S-C 3070 printer and a Bell System Teletypewriter (35ASR), gave a good performance within its design limitations, and very little difficulty was experienced with it.

Type of Queries

Queries submitted by AFLC users will be categorized into two major groups. The groupings are: by the purpose of query, i.e., whether it was submitted from a D041 system user or mission viewpoint, or a technical or data management viewpoint.

A. Listed below are those queries of interest from a D041 system users or mission viewpoint. A number of queries of this type that were attempted are listed regardless of whether they were successfully processed in the ADAM system. An indication made by the user as to the degree of success and to what it was attributed is also given.

1. The first subgroup contains those queries which were used for simulation or extraction of data for determining requirements policies and procedures. About five percent of all queries processed during the experiment were of this type.

(a) A number of queries were processed that simply extract data that could be used in making policy decisions related to the requirements techniques. As an example the base stock level methodology was examined by determining items separately for category I and II, the number of items that have computed base stock levels, and the number of those items requiring manual adjustments to the computed quantities.

(b) Initially, a feature was provided which allowed the production of a requirements computation similar to that performed by the D041 system, but on an item basis (that is, one item was computed through factors, requirements and the RIAR before the next item was commenced) rather than in a batch processing mode. Later, this was changed to a multiple item mode at considerable expense in MITRE time and manpower resources, in that studying and understanding the AFLC D041 system and then developing FABLE routines that provided a similar capability were required. Flexibility and ease of manipulation of the processing and calculations that comprised the requirements computation were provided in the belief that they would provide a tool with which to simulate and test requirements policy and procedures. As potentially useful as these capabilities were, they were used only several times. The inability to use them effectively was attributed to the fact that most operational problems involve information that is not totally available from the D041 data base; in such cases, no complete or meaningful testing or simulation of the problem was

possible. A second situation that was quite prevalent was that, for those problems which could be successfully run, there were no overall management criteria available at the level of the results to determine if the new values obtained were better or worse than those currently used.

(c) A third and larger group of potential queries were of a type that required computations over large segments of the subsets and preferably over the total subsets. Planned use of the system included summarizing dollar information for a complete subset. Examples are:

1. Adjust safety levels of all items in a subset and analyze the impact on gross requirements and net buy requirements.

2. Adjust standard repair cycle for each item in a subset and do same type analysis as above.

The degree of success of these queries was nil because the length of time required to compute requirements for a subset was prohibitive.

Because of this, plans were made and data were provided to place the RIAR files from the AFLC D041 system in the ADAM system. Because of several delays, this feature was not implemented. Even though it would have reduced some of the required computation time, it would still have been prohibitive timewise to process these types of queries. However, they were of the type that could have been very useful tools in making policy decisions. Data had been extracted from another AFLC system, H023 (the RIAR data bank to use as "bench mark" information with which to check the results of these queries. Another simulation exercise of considerable interest, which would have required changing certain factors (due to deployment, etc(, recomputing requirements, buy, retention and disposal, MRS, etc., was planned. Again, because a large number of items were involved, excessive computer time would have been involved in running the simulation so it was not attempted.

The requirements computation routine was considerably improved during the experiment by reducing the translation and processing of one stock-numbered item from forty minutes to twenty minutes. Another great improvement allowed repetitive looping through the routine without requiring retranslation each time. The time required to loop through the routine was reduced to about one minute for each additional stock-numbered item.

(2) The second major subgroup of user queries are those used for extracting data and performing calculations and summarizations for analysis and operational purposes. These queries made up about 40 percent of the total for the experiment. This area of the experiment was relatively more successful than that which obtained data for policy decisions, in that a greater number of this type query was submitted and a larger percent of them was able to be successfully processed.

(a) An example of an interesting and useful application was the extraction of certain forecast data fields to determine if the AMAs had adjusted the requirements data to allow for unusual deployment conditions in the Southeast Asia (SEA) buildup. The queries associated with this application were successfully structured and processed, but only after several attempts. Inconsistent results were initially produced by using the logical operators greater-than-or-equal-to (GQ) and less-than-or-equal-to (LQ) which include null values as well as those that were expected to fall in the indicated ranges of values.

(b) The limiting factors of being unable to summarize large groups of items and not being able to consider data beyond the D041 data base stymied the use of many potentially useful queries. Examples of queries that were not acceptable because of a lack of information are:

1. How many issues were made to a particular base?
2. How many issues were made for war reserve material (WRM)?

(c) Special product listings which selected necessary data from the Techdata and Asset/Usage files were obtained by having MITRE process queries overnight and off-line. The listings were requested for all items in the F^4 subset, and an F105 subset was established at AFLC's request primarily for the purpose of obtaining this special list although the subset was useful for other purposes. The purpose of the lists was to determine if certain items could be maintained effectively at the base level. This was determined by considering the NRTS rate, price, depot repair code, cost category code, along with several other properties.

(d) In order to test the credibility of the data obtained from the ADAM version of D041, a whole class of items, FSC 6720, was printed from the techdata file of the F^4 subset. Among the properties retrieved for each item was that of net buy quantity (NETBUY). The retrieved data which showed no NETBUY was compared to data provided by the AMAs which showed that a NETBUY had been computed for each item. Further checking into the problem did not unravel it; the AMA claimed their records were correct and had been the

ones provided to MITRE. On the other hand, MITRE insisted they had built their files using all, and only all, the data provided by the AMAs. Because of a lack of resources to check into this inconsistency further, a reasonable question as to the validity of some of the output products was raised.

(e) An example of a type of query which had partial success at being answered is: Find the dollar value of usage data (reparable generations and shipments). This query was tried early in the experiment before the price was incorporated in the Asset/Usage file. At that time, only partial success was achieved because of the complexity and time consumption necessitated by cross-file referencing. However, after this query had been brought to our attention, it was attempted again, this time without the necessity for cross-file referencing, since the price had also been incorporated in the Asset/Usage file in the FY 67-1 data base.

(f) A fairly comprehensive series of queries was developed and run against all the available subsets of data by personnel associated with the AFRAMs evaluation

group. The objective was to get statistical distribution of item characteristics (demand rates, percent of base repair, NRTS, condemnation, etc.) by category within each available subset.

The results of the queries were generally successful. Reports containing information from several files were so difficult to achieve because of the constraints of cross-file referencing and 300 character messages, that what should have been one report was often ten separate listings. Most queries were tallies and were normally processed against individual files. To circumvent the problems of cross-file referencing during the analysis, it was proposed that the properties of interest from all the files be combined into one file. This would cause an initial work-load of considerable proportions, but in the long run would probably be the most efficient approach.

B. A large number of queries that ranged from the extremely simple to the very complex were written to test, explore, and probe the FABLE language and the ADAM system. They were primarily submitted by committee members with a technical interest in computers and software. It is estimated that about 55 percent of all queries submitted during the experiment were of this type. Many of the technical observations below were from queries that were originally submitted by mission committee members. A large number of the situations discovered and investigated were as often by accident as by design. Sometimes a query would be designed for one purpose; but through an error an unexpected result or interesting avenue of investigation would open. As a result of these queries, many observations were made concerning the characteristics of the ADAM system and the FABLE language.

Use Observations and Reactions

This section contains observations on the system, how certain features met our needs and how other features were desired. Along with our likes and dislikes are suggestions for improvements.

A. For the FABLE language, the following are observations.

(1) The basic FABLE language was easy to understand and use. The simple retrieval query consists of three parts: the 'for' statement which opens files and selects objects from the files, the boolean statement in which arithmetic operations and logical selection can be performed, and the output statement which allows the results to be saved for future processing, or displayed or printed on various types of output devices in a selected format. Some users were able to learn and write simple working queries in 20 minutes. However, as more complex queries were attempted, adequate documentation to cover the multitude of rules for all possible situations was not available. The rules were unknown to the user and only after several attempts would the requirement or rule become apparent. By monitoring our efforts, MITRE personnel would often be able to set us straight after viewing the results on their monitor station. An example of this was the discovery that floating point numbers could not be compared for equality but had to be tested for being between two limits or in a range of values.

(2) The most complex problems dealt with involved cross-file referencing (relating data in two or more files) for a large number of items. There was a large number of other type queries that seemed quite complex at the time they were attempted, but much of this was due to the lack of operational or working level documentation of the FABLE language. A problem that contributed to the complexity was the lack of a standardized or consistent approach in the construction of statements, and the use of punctuation and keywords.

(a) Cross-file referencing provides an ability to relate data in two or more files. This function was of utmost importance because five files of data were involved in the experiment and many of the queries required data for two or more files. Developing queries for relating data in more than two files was very difficult. For this reason, when multiple files were involved in solving a problem, they were handled two at a time, which was cumbersome for the user. The execution time for one or several objects was reasonable. However, many practical problems required relating data for a complete subset. In fact, the predominant type of query was one which required summarization over a large number of items or a whole subset.

In this situation, the feature is completely inadequate because the processing time for a typical problem was estimated to run into several hours or more. Another limitation caused by this feature was the inability to produce an output report that related data from several files. As an example, instead of one concise report that showed the complete picture, four separate products were prepared that did not provide the desired data relationships. It is highly recommended that an efficient method to relate data in different files be incorporated in future systems.

(b) The lack of standardization in construction of statements and in the use of punctuation and keywords made language difficult to learn and use. Although the items mentioned in this section are not individually of great consequence, cumulatively they do lend difficulty to the use of the language.

1. It is believed that the selection of system keywords could have been more meaningful. A case in point is the use of the keyword "all" in several different contexts. It is used

in string substitution in the following context, "LET DATE MEAN (31 AUGUST 1966) FOR ALL USING RESCAN". Here "ALL" refers to all input/output devices. In the retrieval statement "FOR TECHDATA 16306762128, PRINT ALL.", "ALL" refers to all properties in the techdata property list. Another use of "ALL" is "FOR ASSETUSG 161O1829917. ALL (ATYPE SAFETY LEVEL LS 10 CHANGE ATYPE SAFETY LEVEL TO 10)", which causes stepping through all repetitions until the boolean is evaluated as false. The use of "NOT ELSE" to create a null file is another example of words with unusual and special meanings. An example of its use is, "FOR TECHDATA. NOT ELSE, SAVE ALL. NAME NEWFILE." A portion of the TALLY PRINT statement is redundant in that it must be written as "....PRINT TALLY TALLY...."

2. The statement of some queries required awkward construction. For instance, the "UNTIL" feature which limits output only

and is not functionally related to the boolean clause, is connected to the boolean clause, but is separated by a comma from the output phrase to which it applies. When data is sorted in conjunction with a query, the sort is performed before the printing, yet the print statement precedes the sort phrase.

3. The requirement to use commas to separate certain items such as objects and properties and the requirement that they not be used to separate other items such as device names was confusing. The requirement for a comma preceding an output statement, and the requirement for no comma before a change phrase, was confusing in that one or the other of the two phrases can occupy the same position in the construction of a query.

4. It is recommended that unique and meaningful system keywords be selected and that consistent punctuation and construction be used.

(c) A fair amount of confusion resulted before the characteristics of floating point representation were understood. As an example, several attempts to find values known to be in a file ended in failure when it was found that an "equal" test was not valid and that a range test then had to be performed. The method the system uses in handling floating point numbers caused confusing results in a number of test calculations that were run. Another idiosyncrasy was the erratic placement of the decimal point in the scientific notation of numbers. Under various circumstances, it would be placed either to the left or right of the first significant digit. Although not an error, it would be good practice to place the decimal consistently to the right of the first significant digit to aid in the readability of the numbers and reduce reading errors. It is recommended that the problems associated with converting and using floating point numbers be resolved before using in an operational management system. It is believed that most managers would prefer to use regular numbers rather than scientific notation because they are accustomed to the regular numbers and find them easier to work with.

(d) The solution of a number of the problems submitted became complex because of the sheer magnitude of the number of queries involved to obtain a solution. It was common to have from four to six queries or more to obtain a problem solution. Several problems were not even attempted because of the excessive amount of work that would have been involved. The use of the string substitution feature, which allows one keyword to replace a longer phrase or statement, was useful in reducing the number of queries and otherwise simplifying and making them more readable. The teletype constraint of 300 character messages limited the amount of information that could be placed in one substitution. However, the basic difficulty was caused by the necessity for a linear lay-out of the problem, which is inherent to the system. By not being able to use recursive routines to set up a small problem and loop through it automatically, a tremendous amount of time was required to set up counters and files and individual instructions that reference all the data individually. It is recommended that some method be found to allow recursive looping through a shorter routine.

(3) A number of desirable features were incorporated in the FABLE language.

(a) The wide range of functions allowed data in the ADAM files to be retrieved, changed, and deleted. New data was added and new files were defined in terms of an existing one and created. Types of information that are repeated over a number of times for one object were handled easily by the use of the "repeating group." The boolean expressions allowed complex logical filters to be constructed. As mentioned before, string substitutions were very useful. Although not used much, the synonym capability would be very useful in an operational system.

(b) A macro instruction capability was provided with the language that permitted a variety of interesting queries without the necessity for learning a great deal about the intricacies of the FABLE language. These instructions were used by only inserting the required parameters. The routines that were developed to compute factors, requirements and RIAR data were of this type. Other examples of macro functions are TALLY, POWER and SQUARE ROOT. Of these, the tally function was the most used by far. Because of its flexibility, it has replaced several of the

previously used macros, and was frequently used in lieu of preparing a FABLE statement. It provides for summarizing the number of times each property value appeared in the data individually or in combination with another property value. Logical operators were used in the tally to group numerical values into ranges. A shorthand method of defining the ranges was provided by allowing for the starting value, the increment, and ending value. This made it easier and faster to use. Although the tally function was very useful, it had several limitations. Because of its ability to tally on only 25 values, it could not be used to solve several mission queries. One of these queries required tallying on 300 values. Another limitation was that logical values could not be tested for other than equal and unequal conditions. Quite often situations arose in which it would be necessary to tally a number of properties over the same range of values. The tally function can only handle one property at a time. This reduces the power of the function, particularly when the tally is used in conjunction with a complex boolean statement. In one situation, instead of one query,

eight tallies and associated booleans were required. This nearly caused writer's cramps. It is recommended that the limitations mentioned above be removed from the tally function and that similar constraints be removed from other functions. This would make a system much easier to use.

(c) Documentation of the FABLE language was very good from an academic viewpoint. Several reports on the basic language provided adequate details for training purposes but had inadequate details from an operational viewpoint, in that many rules of structure and punctuation of statements and the detailed functions of a statement were not included in this documentation. Those functions that have limited uses should be given a title that describes these limitations. A set of operating documents that indicates what happens under all conditions should be provided. The conditions that cause errors should be shown with the error and how it can be corrected if possible. Also the conditions that cause a system failure should be documented. Without this documentation, a person could easily forget some of

the conditions that would cause a failure and thereby unintentionally contribute to causing one.

B. A number of queries run during the experiment gave an insight into some ADAM characteristics. The most significant of these are discussed below.

(1) Although there was no special attempt to measure precisely the efficiency of translation and execution of queries, observations indicated that in a gross manner the translation time is quite large in comparison to execution time. For instance, one routine that took 20 minutes to translate ran in about one minute. Another routine that took six minutes to translate ran in 15 seconds. It is doubtful that an operational system could afford to translate and retranslate each query. Once a query has been translated, it should be placed in a routine library if there is a reasonable chance it could be used again.

(2) The lack of a checkpoint restart feature is a serious problem, particularly in light of the many things which cause a system failure. Although a restart feature is provided, it is essentially a tape load that starts the system over. It does not provide for saving string substitutions or changes to the data base, so every time a system failure occurred, considerable processing time was lost. A checkpoint restart is needed.

(3) The input priority scheduling feature was found to be inconvenient. All queries were placed in a queue on a first-come, first-serve basis until the queue of some items was full, after which additional queries were lost. As a minimum, a message should notify the user that the queue is full. Preferably, the input message storage area should be large enough so that a user would not have to concern himself with the system not being able to accept his input. Also, it would be very desirable to have a priority query that could take its place at the head of the queue for immediate processing.

(4) Error messages should be accurate. Specific error messages are generated for conditions that require a general statement of an error. A good deal of confusion resulted when users got specific messages which they took literally. For instance, one error message stated "a comma was missing" when obviously there was none missing. The error message should be stated in general way unless a specific error has been detected.

(5) The separate definitions and characteristics of logical, floating point, and integer values place excessive restrictions on the use of the system. For instance, it would have been desirable to have been able to compare logical values on other than equal/unequal. There are certain restrictions on the use of the tally

function and floating point numbers which limit their full potential. A system designed for user flexibility should allow any meaningful operation on all types of data.

(6) The inability to nest file definitions causes a necessity for redundant storage of each part that is to be defined. Storage space could be saved if nested file definitions were accepted.

(7) Below are some observations concerning the remote station equipment and operation.

(a) There is a need for status indicators at the remote station to show the current status of the equipment (central processor), software system (ADAM), and the data or telephone lines. Several times the sytem or equipment failed or the lines were out, but the problems were unknown to the remote user who kept trying to input more queries.

(b) Sometimes, based upon receiving the results of a query, it is unnecessary to process another one already in the input stack. For this reason, it would be desirable to be able to delete queries from the stack.

(c) The message number and a message trail could be typed out in all circumstances for checking purposes, and the complete message provided upon request.

(d) It was sometimes difficult to relate answers to the appropriate query otherwise identify the product. To alleviate this, all output information should have the date and time prepared, a title, the subset involved, the requestor's name, and and identification number relating an answer to the query that generated it.

(e) The capacity for making changes and corrections to queries being keyed into the system would be necessary in an operating system. Considerable time is lost in starting queries over when the error is not detected when it was made. A method of using a grid of lines and columns with the necessary equipment and software could allow one to identify the position of the error so that it could be located and corrected even after it had been input to the input stack.

(f) A method is needed for terminating output functions on the printer. If a query is mistakenly written which produces such a volume of output that it would take 20 minutes or even several hours, there is no recourse to waiting for the printing except to stop the job and restart. Several times during the experiment the run

had to be terminated to stop printing.

(g) The necessity for knowing the objects in a roll before querying them is an inconvenience. When an object is not in a roll, a message indicating a null value would be more appropriate than an error message.

C. Just about everyone on the ADAM committee initially agreed to the desirability of the objectives of a user-programmed, on-line system. Some thought that it was redundant even to test the technique and that a need for a similar, but more advanced, operating system existed and should be obtained. It was pointed out that, although the objectives were without question desirable, the purpose was to find if they would work in the AFLC environment, and to evaluate the ADAM implementation of the technique. This was the approach taken and below is a summarization of the users' evaluation.

(1) An ability to obtain an answer to a query in a short time frame was the single most impressive aspect of the system for many of the users. Because of previously mentioned problems, only rather simple queries were answered in a time span representative of on-line operation, but even those that took longer, several days to a week, were faster than any other method that could have been used to get the data, if it could have been obtained at all by other means. Having the ability to acquire data quickly as it is needed is a prime requisite for any new system design approach. It was generally agreed by committee members that a reaction time of 24 hours would be adequate for most

system responses, but that a priority response of an hour or preferably less would be desirable in some situations.

(2) At first, it seemed desirable to have the users program or state their own queries in the FABLE language. However, it was found that the mission personnel were not inclined to use the language to structure queries. Although it is an English-like syntax language, the rules of syntax are much more restrictive than natural English. In reality, FABLE is a programming language. Even though it is a high-level language, there are a number of precise rules to be remembered, as there will be with any language in the foreseeable future. The mission personnel preferred not to learn them, but to allow professional programmers to do the job. To effectively utilize the system, it was found that a user must be more proficient than could normally be expected of him. Therefore, from an efficiency viewpoint, it was desirable to have a highly trained programmer translate the problem into the FABLE language and operate the remote equipment.

(3) It was found that many mission personnel, even though they are experts in their respective areas, are not good system analysts, in that they would not state the problem precisely enough even in English to get the solution they intended. This lends more weight to the argument for having an intermediary who is an expert

analyst programmer between the ultimate user and the system.

(4) A considerable amount of time can be spent even by a person reasonably familiar with the query language. As an example, an hour could be spent formulating the problem solution in English. An hour or more could be spent in converting this into FABLE. Then another half-hour could be spent keypunching an instruction tape. If a considerable amount of summarization and cross-file referencing is involved, the query can run from 20 minutes to several hours.

(5) Through the use of string substitutions, normal English-looking statements can be built. On a test basis, some of these were prepared. But because it is human nature to take a shortcut, practically all statements written by both AFLC and MITRE personnel were so highly abbreviated that they looked as unintelligible to the uninitiated as any other programming language. The same is true of COBOL; it appears nearly hopeless to try to get people to write a fully intelligible statement. What is needed possibly is a method by which the user inputs a highly abbreviated form; but for documentation and recording of the results the system outputs a full English-like statement. Some aspects of this are already in ADAM in that when a macro-type routine or string substitution is used, a full statement of the routine or statement is printed on the output. This would have to also be extended to data

properties, file names, and certain key words to have really readable product.

Summary Recommendations

A. A composite list of the detailed recommendations are given below:

(1) The cross-file referencing function should be made more efficient.

(2) Unique and meaningful system keywords and consistent punctuation and construction should be used throughout the language.

(3) A method to allow looping through a shorter routine should be developed in lieu of laying a long problem out in a linear fashion.

(4) Limitations on certain functions that allow them to handle only one object at a time should be removed. Likewise, limitations on the various types of data should be removed so as to allow any meaningful operation on all data.

(5) Detailed operational documentation is required. This should include conditions that cause errors and system failure, as well as error correction routines.

(6) If possible, translation time should be reduced. In any case, the translated queries that have a chance of being used again should be saved.

(7) A checkpoint restart would be needed in an operational system.

(8) Error messages should be stated accurately.

(9) Method is needed for terminating output.

(10) A provision for a method of relating answers to the appropriate query should be made.

(11) In an operational system, the ability to correct queries after they had been input to the system would be desirable.

B. Projecting some requirements for a new operational design, it was evident from this experiment that a large majority of the queries required computations and summarizations over an entire subset of data. For the simple direct queries addressed to data in only one file, acceptable retrieval times from several seconds to several minutes were registered. When a query, that would be of a relatively common type in an operational system, required data from several files, the complexity of the queries increased appreciably, but the execution time became extremely prohibitive, resulting in processing times of twenty minutes to estimates of several hours. The above tests were small indeed compared to what is being envisioned for AFLC's next major system design effort. The average of 2000 items in a subset makes it about one forty-second as large as the total D041 data base. The integrated system design being planned incorporates many systems and their data. This could add a total of 10 times the volume of the D041 system alone. That would amount to over 400 times the data in the subsets used in the experiment and would require several billion characters of mass storage to put all of it on-line at one time. What would likely be more practical is a combination of tape and disk files. What is needed is a detailed concept of operation, and design features for a system that can handle a problem

of this magnitude. It would likely contain both serial and random processing features and many of the generalized data management characteristics, if it can be effectively adapted to handle problems of this magnitude.

C. It is recommended that organizations with resources, talent, and inclination attack the problem of improving the operating characteristics of generalized data management and its supporting techniques of on-line time-sharing, remote operation for extremely large data bases of the order mentioned above.

D. It is recommended that the AFLC data management organization proceed with definite plans to study and convert or otherwise acquire new state-of-the-art techniques that would be applicable to AFLC's special requirement of extremely large data bases.

E. The mission planners should be aware of these new techniques and plan, when applicable, for their inclusion in systems they redesign.

Achievement of Objectives

A. The first objective,"test, verify, and demonstrate the technology of the ADAM system", was an ESD/MITRE objective and was fulfilled through performing the experiment.

B. "Refining the ADAM technique" was entirely an ESD/MITRE objective; however, AFLC personnel contributed to its achievement by providing observations and recommendations on the system to ESD/MITRE.

C. "Determining the potentialities and deficiencies of generalized data management" was an objective of ESD/MITRE and will be covered in their comments. A similar objective for AFLC is stated in D. from a different viewpoint.

D. The main AFLC objective was to determine if the ADAM technique is applicable to AFLC logistic systems. The detailed and summary recommendations contain many observations which indicate to what extent the technique is applicable; therefore, this objective has been achieved.

Conclusions

The concept of generalized data management has been examined in detail from a user viewpoint through this experiment. Some of the techniques are in such a state of development that they could be used in AFLC systems. The large volume of data in AFLC files makes the operating efficiency of a system of prime importance. Because a generalized system by nature is less efficient than one of specific design, it will be necessary to build large integrated systems using specific design criteria and to set the bounds or framework within which flexible generalized functions can be used efficiently.

Some specific conclusions are:

A. At the operating level, it highly desirable to have a capability to obtain data when needed. The requirement for response time will vary from one situation to another. However, for working continuity, at least a twenty-four hour response time is needed. Although some would like immediate response of a few seconds for certain applications, there would normally be very few situation in which an hours response would not be adequate. This criterion would not require a complete on-line approach but would allow a combination of serial and random on-line processing.

B. A raft of problems plagued the use of English-like customer-oriented, syntax language by mission personnel. The language still requires a formal structure that must be learned, and is quite complex when all of its features are used. It takes a fair amount of time to convert an English statement of the problem into FABLE, and it is nearly impossible to get people to write full English statements that are understandable to someone else. The English-like syntax language has potential for use by specialists, but does not seem too promising for the "man-of-the-street."

C. The overhead cost of a generalized system like ADAM is high; but through judicious selection and design, some of its more important benefits could be incorporated in an operating system at reasonable costs.

D. It was found that programming time varied considerably from one application to another, depending on whether the particular problem used efficient or inefficient features of ADAM. When efficient features were used, its generalized nature provided a solution much faster than could be obtained by conventional programming methods. However, when the nature of the problem required the use of inefficient features, it was estimated to take as long or longer than conventional programming methods.

E. The experiment has given AFLC personnel an opportunity to examine in detail the concepts and actual operation of many state-of-the-art techniques. Some of these are: generalized data management, on-line, time-sharing, remote station operation using an English-like user syntax language. Having this experience puts us in a much better position for designing improved systems. By comparing the

results of the experiment to the objectives, the experiment was viewed as a success. However, the final measure of success will be achieved if the detailed comments or summary recommendations derived from them help guide someone in designing and developing a better system.

USAGE ESTIMATES

In order to provide the reader with an order of magnitude, some usage figures were collected and are presented here. Although the duration of the whole experiment might be considered a learning period, no figures are given for April 1966 because it might be considered an indoctrination period. At the time of writing, August figures were not available and are only estimates.

From 1 April 1966 until the switchover to 67-1 data bases, the F106 subset was used almost exclusively, mostly due to its convenient size. As each new 67-1 subset became available, it was used until the next was ready. Ultimately, the most frequently used subsets were those based on B52, F4, and F105.

Table IV presents the volume usage of the remote operation. The first line figures do not include equipment checkout operation. They do include startup and abnormal reload time. The volume figures do not include ADAM utility messages such as $TELL or $TIME. The last line figures represent queries which did not end by error returns from the ADAM system.

Table IV

Usage of Remote Operations

	MAY	JUNE	JULY	AUGUST
Number of hours of ADAM operation	14.6	12.4	21.3	12
Number of days	19	16	20	7
Average no. of queries per hour	43	43	47	50-55
Average no. of queries from AFLC per hour	25	17	31	40-45
Percentage successful AFLC queries	64	69	70	85-90

Based on sample data collection, some estimates are available about the type of queries generated by AFLC. Only the successful ones were categorized. These classifications are broad and make no attempt to relate the queries to the mission.

Type	Percent of Successful Dayton Queries
Retrieval	40
String Substitution and Execution	45
Macro-queries	10
Computation	5

SECTION VI

CONCLUSION

The project objectives listed in the Introduction to this report were attained at varying levels of accomplishment and with varying degrees of success. The generalized data management concept was demonstrated in an operational-type application. Specific areas of the generalized data management concept were identified for consideration in future implementations of the concept. Users were introduced to and educated in the concept's potential, although there were fewer active on-line users than had been anticipated. Finally, the ADAM-based Requirements System did serve as a vehicle for testing certain logistics management methods, even though many policies could not be tested because of limitations on data base contents or size and/or machine processing times.

Measuring the degree of success in reaching the latter two goals is discussed in Section V. In attempting to fulfill the first two goals, the following general observations are made, based on the experiences gained by the designers, programmers, and users of the ADAM-based Requirements System. Some appear to be obvious to experienced programmers, and the experiment merely served to verify them.

Programming techniques based on the generalized data management concept can be useful when properly applied in an operational environment such as exists at the Air Force Logistics Command. In particular, the most useful portions of the generalized concept as exercised by the users were:

(a) the capability of describing and introducing new data bases to the computer system comparatively easily and quickly;

(b) the capability of extracting information from the data base; and

(c) the capability of performing relatively simply computations and tallies on selected portions of the data bases.

Certain generalized data management techniques as implemented in ADAM, such as random-access to data and an on-line, interpretive mode of operation, cannot completely replace serial batch-processing techniques in large scale management information systems such as exist at AFLC in the near future. In particular, data systems which are required to produce voluminous reports based on computations involving a high percentage of the data elements in a large data base should be programmed using serial, batch-processing techniques because of the relative efficiency of these techniques in terms of computer-processing time.

In view of the above conclusions, future designers of management information systems should consider implementing the applicable portions of the generalized data management concept with provisions for using batch-processing techniques on the common files in a compatible fashion.

User personnel with programming experience find that a higher order, near-English computer language such as FABLE is relatively easy to use provided the constraints are well documented. Nonprogrammer-users such as are in the mission elements at AFLC are not used to express their ideas following such rigid rules and,

therefore, have trouble in learning to write other than the most trivial queries. Furthermore, some of the nonprogrammer-users believe that it should not be a part of their job to write queries, no matter how simple the language may be. Thus, systems should not be designed with the expectation that the ultimate user of the product will do his own programming unless its languages are easy to use by having it especially tailored in the vocabulary and function of the application needs, or the class of users is limited to programmers or personnel readily adaptable to such programming tasks. Further work in developing systems that are easy to use is needed.

Specifying and collecting the most suitable raw data and determining the appropriate level of aggregation is difficult, but it is a crucial part of designing effective experimental or operational management information systems. Thus, this area should not be neglected. Efforts should be made to obtain the best data for the job rather than just accepting that which is readily available.

APPENDIX I

DESCRIPTION OF IBM 7030 AND IBM 7080 COMPUTER CONFIGURATIONS

The table on page 90 describes the IBM 7030 configuration at the ESD/MITRE System Design Laboratory and the IBM 7080 configuration used by AFLC.

DESCRIPTION OF IBM 7030 AND IBM 7080 COMPUTER CONFIGURATIONS

	IBM 7030	IBM 7080
Core Memory Size	65,546 words (64-bit-520,000 characters (8-bit))	160,000 characters
Input/Output	Basic Exchange - 16 channels 3 Channels for 12 Tape Drives 4 Channels Printer Card Reader Card Punch Operator Console 7 Channels for Teletype and Phone Lines 1 Channel for Display (cathode-ray) consoles 1 Channel for Stromberg-Carlson Printers	3 Banks for Central Storage (0,1,3) 2 Banks for Communications (2,4) Bank 2 - 4 Channels to 2 Tape Control Units 7621 Bank 4 - Up to 6 Channels to IBM 7908 Data Channel 7908 Data Channel, in conjunction with Bank 4, provides up to 6 additional I/O Channels. Two of these are high speed channels.
Tape Drives Density Options	12 729 Mod IV 200 or 556 cpi	20 729 Mod VI 200,556,800 cpi Speed: 112.5 ips
Disk	32 Million Characters (8-bit) or 4 Million 64-bit words Rotation Time - 33 millisec. Transfer Time - 4 microsec.	

APPENDIX II

BACKGROUND INFORMATION CONCERNING SOFTWARE

One of the biggest problems associated with the automation of management information gathering and decision making processes is the inherent inflexibility of the programs used to describe the processes to the computer. The procedures must be described following a rigid set of rules that are difficult to learn and apply. The bookkeeping problems associated with the application of the rules increase rapidly as the number of instructions in the computer program increases. Thus, the costs of creating and changing large computer programs are high, and many of the capabilities of the computer are not exploited.

The technique used by computer programmers to reduce the complexity of the programming problem is to use the computer to do much of the dogwork necessary to adapt a statement of the problem solution procedures to the machine. Computer programs written for the purpose of aiding programmers in this manner are known as software. (The term software is often used to describe all computer programs and documentation, but this is not the connotation implied here.)

Software exists at many different levels of sophistication and generally addresses specific classes of problems. For example, translators, the computer programs used to convert statements of procedures expressed in a language convenient for the programmer to a form usable by the machine, can be broadly categorized according to their sophistication as assemblers, compilers, or interpreters.

Assemblers produce only one machine-language instruction for each programming (source) language instruction, but handle the

bookkeeping required to assign specific addresses in the computer's memory for the program and data elements. Because of the one-to-one correspondence between the source and machine languages, all of the capabilities of the computer are still available to the assembly-language programmer, but he must use considerable diligence and skill in constructing nontrivial programs.

The compiler produces more than one machine-language instruction for most source-language instructions. Programmers find that compiler languages are easier to use and to learn to use than assembly languages. However, the languages that are easiest to use tend to offer the least versatility to the programmers. Furthermore, the machine language instructions (object code) produced by a compiler are often inefficient in their use of the computer's memory and in the computer time required to run the program.

Because of the lack of versatility of compilers, they generally are designed to handle a specific class of problems best. FORTRAN, for instance, addresses scientific and engineering type problems that could be described in a language similar to algebra. Another compiler language, COBOL, is designed to handle business data processing problems.

The interpreter is similar to a compiler in that the ratio of object code to source language is greater than one. However, it differs in its mode of operation. A compiler produces all of the machine code required by the program before the problem program starts to run. An interpreter, on the other hand, causes each independent block of the machine-language instructions produced to be run before translating the next block. Thus, interpreters are used in on-line systems such as in command and control

applications where it is important for the programmer or operator to be able to interact with the system by observing intermediate results and making required changes as the program is being executed.

There are other types of software available to aid programmers. These include executive and monitor routines to handle the control and scheduling of jobs for the appropriate components of the computer in processing a program, and debugging routines to aid in finding program errors. When executives, monitors, interpreters or compilers, debugging routines, etc., are combined in an integrated system, the result is often called a programming system.

Large scale management information systems are among the most costly to program and reprogram. Because the Air Force has built and is building many such systems, a reduction in the costs of producing and modifying such systems is very desirable. The development of programming systems based on a concept known as generalized data management is a step in that direction. Two existing systems based on that concept are the Advanced Data Management (ADAM) System developed by the MITRE Corporation, and the LUCID System developed by the System Development Corporation. More information on these systems and the generalized data management concept is contained in Reference 17.

APPENDIX III

DATA BASE FILE SIZES

67-1 File Sizes

F4	FUTPROG	PASTPROG	ASSETUSG	TECHDATA	APPL	TOTALS
No. of Objects	218	169	1378	1389	1386	
Max. Object Size (arcs)	1	1	1	1	4	
File Size (arcs)	43	32	179	362	145	
Rolls Size (arcs)	5	5	19	21	22	
Roll PROP (arcs)						3
Total Size (arcs)	48	37	198	383	167	836
Input Volume (no. Rcds)	53	44	805	279	292	1473
File Generation Time (min.)	9	7	91		31	138
Processor Time (min.)	4	3	31	48	23	109

F105	FUTPROG	PASTPROG	ASSETUSG	TECHDATA	APPL	TOTALS
No. of Objects	305	254	503	1149	1136	
Max. Object Size (arcs)	1	1	1	1	8	
File Size (arcs)	61	48	154	299	247	
Rolls Size (arcs)	6	5	8	19	20	
Roll PROP (arcs)						3
Total Size (arcs)	67	53	162	318	267	867
Input Volume (no. Rcds)	95	67	412	231	407	1212
File Generation Time (min.)	9	8	81		44	142
Processor Time (min.)	5	4	25	42	36	112

FSC12	FUTPROG	PASTPROG	ASSETUSG	TECHDATA	APPL	TOTALS
No. of Objects	Ø	94	2763	2848	2839	
Max. Object Size (arcs)	1	1	1	1	1	
File Size (arcs)	2	19	714	741	206	
Rolls Size (arcs)	2	4	36	38	37	
Roll PROP (arcs)						2
Total Size (arcs)	4	23	750	779	243	1801
Input Volume (no. Rcds)	Ø	24	1907	570	453	2954
File Generation Time (min.)	3	5	122	109	48	287
Processor Time (min.)	Ø	4	115	94	32	245

MMC	FUTPROG	PASTPROG	ASSETUSG	TECHDATA	APPL	TOTALS
No. of Objects	268	96	627	2520	2456	
Max. Object Size (arcs)	1	1	1	1	3	
File Size (arcs)	36	22	164	656	294	
Rolls Size (arcs)	6	4	11	34	34	
Roll PROP (arcs)						3
Total Size (arcs)	42	26	175	690	328	1264
Input Volume (no. Rcds)	54	32	459	506	499	1550
File Generation Time (min.)	6	6	32	113	46	203
Processor Time (min.)	4	2	27	100	36	169

B52	FUTPROG	PASTPROG	ASSETUSG	TECHDATA	APPL	TOTALS
No. of Objects	403	260	3313	3324	3308	
Max. Object Size (arcs)	1	1	1	1	4	
File Size (arcs)	68	48	895	864	481	
Rolls Size (arcs)	7	6	42	44	46	
Roll PROP (arcs)						3
Total Size (arcs)	75	54	937	908	529	2506
Input Volume (no. Rcds)	91	69	2373	666	1125	4324
File Generation Time (min.)	9	8	153	121	99	390
Processor Time (min.)	7	5	141	105	82	340

66-2 File Sizes

F106	FUTPROG	PASTPROG	ASSETUSG	TECHDATA	APPL	TOTALS
No. of Objects	204	209	1227	1226	419	
File Size (arcs	40	117	388	299	37	881
Rolls Size (arcs)	5	5	17	22	19	68
Roll PROP						3
Total F106 File (arcs)	45	122	405	321	56	952
No. of Physical Records	57	126	1017	481	150	1831
File Generation Time (min.)	14	25	142	184	69	433
Processor Time (min.)	9	21	136	175	59	
10 APPLICATION	FUTPROG	PASTPROG	ASSETUSG	TECHDATA	APPL	TOTALS
No. of Objects	201	215	771	773	453	
File Size (arcs)	40	119	365	259	57	840
Rolls Size (arcs)	5	5	12	17	15	54
Roll PROP						3
Total 10 Application File	45	124	377	276	72	897
No. of Physical Records	58	129	860	542	241	1830
File Generation Time (min.)	10	27	131	136	71	375
Processor Time (min.)	8	22	126	124	62	

26 APPLICATION	FUTPROG	PASTPROG	ASSETUSG	TECHDATA	APPL	TOTALS
No. of Objects	238	246	2365	2362	453	
File Size (arcs)	43	129	767	670	50	1659
Rolls Size (arcs)	5	5	31	38	30	109
Roll PROP						3
Total 26 Application File	48	134	798	708	80	1771
No. of Physical Records	63	143	2034	1255	207	3702
File Generation Time (min.)	10	27	285	559	184	1065
Processor Time (min.)	7	24	275	544	168	

APPENDIX IV

PROPERTIES OF D041 FILES

PROP NAME	TYPE PROP	DESCRIPTION	RG NAME	LEVEL	FILE NAME
OBJECT NAME		APPLICATION			FUTPROG
SERV	L	SERVICE CODE	FUTCFM	1	FUTPROG
RCDID	L	RECORD IDENTITY	FUTCFM	1	FUTPROG
PC	L	PROGRAM CODE	FUTCFM	1	FUTPROG
PRGDOCUM	R	PROGRAM DOCUMENT	FUTCFM	1	FUTPROG
HQCD	L	HEADQUARTERS CODE	FUTCFM	1	FUTPROG
BYR	I	BEGIN YEAR	FUTCFM	1	FUTPROG
BQTR	I	BEGIN QUARTER	FUTCFM	1	FUTPROG
AMA	L	HOME AMA	FUTCFM	1	FUTPROG
RETENT	I	RETENTION	FUTCFM	1	FUTPROG
FDATE	I	FUTURE DATE	CQ	2	FUTPROG
FQUANT	I	FUTURE PROGRAM QUARTERLY QUANTITY	CQ	2	FUTPROG
AASERV	L	AIRBORNE ALERT SERVICE CODE	FUTAA	1	FUTPROG
AARCDID	L	AIRBORNE ALERT RECORD IDENTITY	FUTAA	1	FUTPROG
AAPC	L	AIRBORNE ALERT PROGRAM CODE	FUTAA	1	FUTPROG
COSTCAT	L	COST CATEGORY	FUTAA	1	FUTPROG
AADOCUM	R	AIRBORNE ALERT PROGRAM DOCUMENT	FUTAA	1	FUTPROG
HQCD	L	HEADQUARTERS CODE	FUTAA	1	FUTPROG
LTQ6	I	LEADTIME QUANTITY 6 MO.	FUTAA	1	FUTPROG
SLRC6	I	STK LEVEL/REPAIR CYCLE 6 MO.	FUTAA	1	FUTPROG
LTQ9	I	LEADTIME QUANTITY 9 MO.	FUTAA	1	FUTPROG
SLRC9	I	STK LEVEL/REPAIR CYCLE 9 MO.	FUTAA	1	FUTPROG
LTQ12	I	LEADTIME QUANTITY 12 MO.	FUTAA	1	FUTPROG
SLRC12	I	STK LEVEL/REPAIR CYCLE 12 MO.	FUTAA	1	FUTPROG
CYCDAT	I	CYCLE DATA	FUTAA	1	FUTPROG

OBJECT NAME		MASTER STOCK NUMBER			APPL
MSNTD	L	MASTER STOCK NUMBER TECHDATA		M	APPL
MSNAU	L	MASTER STOCK NUMBER ASSET USAGE		M	APPL
APPLNO	L	APPLICATION NUMBER	APPLNOG	1	APPL
APNOPP	L	APPLICATION NUMBER IN PASTPROG	APPLNOG	1	APPL
APNOFP	L	APPLICATION NUMBER IN FUTPROG	APPLNOG	1	APPL
PSC	L	PROGRAM SELECT CODE	APPLNOG	1	APPL
RESTRICT	I	PROGRAM RESTRICTION CODE	APPLNOG	1	APPL
NEWFLAG	L	NEW/CHANGED APPLICATION FLAG	APPLNOG	1	APPL
QPA	I	QUANTITY PER APPLICATION	APPLNOG	1	APPL
PASTPC	F	PAST APPLICATION PERCENT	APPLNOG	1	APPL
FUTPC	F	FUTURE APPLICATION PERCENT	APPLNOG	1	APPL
ERRC	L	ERRC	APPLNOG	1	APPL
DDC	L	DEFERRED DISPOSAL CODE	APPLNOG	1	APPL
FYR	I	FACTOR YEAR	ITEMPP	2	APPL
FMC	I	FACTOR MONTH	ITEMPP	2	APPL
QPFACTOR	F	QPA TIMES PASTPC	ITEMPP	2	APPL

OBJECT NAME		MASTER STOCK NUMBER		M	ASSETUSG
PRICE	F	PRICE		M	ASSETUSG
ACCT	L	ACCOUNT CODE	ATYPE	1	ASSETUSG
CYCLEYQ	I	CYCLE DATE YEAR	ATYPE	1	ASSETUSG
SBDA	I	SERVICEABLE BASE DEPOT ASSETS	ATYPE	1	ASSETUSG
SCONT	I	SERVICEABLE CONTRACTOR	ATYPE	1	ASSETUSG
BUA	I	UNSERVICEABLE ASSETS BASE	ATYPE	1	ASSETUSG
DUA	I	UNSERVICEABLE ASSETS DEPOT	ATYPE	1	ASSETUSG
CUA	I	UNSERVICEABLE ASSETS CONTRACTOR	ATYPE	1	ASSETUSG
TOCA	I	TOC ASSETS	ATYPE	1	ASSETUSG
WORKA	I	WORK ORDER ASSETS	ATYPE	1	ASSETUSG
WRMA	I	WRM ASSETS	ATYPE	1	ASSETUSG
SAFETY	I	SAFETY LEVEL	ATYPE	1	ASSETUSG
DOTM	I	DUE OUT TO MAINTENANCE	ATYPE	1	ASSETUSG
SINTRAN	I	INTRANSIT SERVICEABLE ASSETS	ATYPE	1	ASSETUSG
UINTRAN	I	INTRANSIT UNSERVICEABLE ASSETS	ATYPE	1	ASSETUSG
SADD	I	ADDITIVE SERVICEABLE ASSETS	ATYPE	1	ASSETUSG
UADD	I	ADDITIVE UNSERVICEABLE ASSETS	ATYPE	1	ASSETUSG
RPTNO	I	NUMBER OF BASES REPORTING	ATYPE	1	ASSETUSG
NAME		RECORD IDENTITY B-I	USAGE	1	ASSETUSG
ZEROTAL	I	ZERO TALLY MOS	USAGE	1	ASSETUSG
UBYR	I	USAGE BEGIN YEAR	USAGE	1	ASSETUSG
UBMO	I	USAGE BEGIN MONTH	USAGE	1	ASSETUSG
UEYR	I	USAGE END YEAR	USAGE	1	ASSETUSG
UEMO	I	USAGE END MONTH	USAGE	1	ASSETUSG
UQUANT	I	USAGE QUANTITY	UQ	2	ASSETUSG
UYR	I	USAGE YEAR	UQ	2	ASSETUSG
UMO	I	USAGE MONTH	UQ	2	ASSETUSG
NAME		RECORD IDENTITY L-W	ADDAS	1	ASSETUSG
ABYR	I	ADDITIVE BEGIN YEAR	ADDAS	1	ASSETUSG
ABMO	I	ADDITIVE BEGIN MONTH	ADDAS	1	ASSETUSG
BALANCE	I	BALANCE	ADDAS	1	ASSETUSG
ADQUANT	I	ADDITIVE QUANTITY	ADQ	2	ASSETUSG
AYR	I	ADDITIVE YEAR	ADQ	2	ASSETUSG
AMO	I	ADDITIVE MONTH	ADQ	2	ASSETUSG
DATE	I	CYCLE DATE YEAR, QTR	ZTYPE	1	ASSETUSG
ASDEP	I	DEPOT	ZTYPE	1	ASSETUSG
ASCONUS	I	CONUS BASE	ZTYPE	1	ASSETUSG
ASOS	I	OVERSEAS BASE	ZTYPE	1	ASSETUSG
ASCONT	I	CONTRACTOR	ZTYPE	1	ASSETUSG
ASWRM	I	WAR READINESS MATERIAL	ZTYPE	1	ASSETUSG
SXAAD	I	SERVICEABLE EXCESS ASSETS DISPOSABLE	ZTYPE	1	ASSETUSG
UXAAD	I	UNSERVICEABLE EXCESS ASSETS DISPOSABLE	ZTYPE	1	ASSETUSG

OBJECT NAME		APPLICATION			PASTPROG
RCDCODE	L	RECORD CODE	PPQUANT	1	PASTPROG
SERV	L	SERVICE CODE	PPQUANT	1	PASTPROG
PC	L	PROGRAM CODE	PPQUANT	1	PASTPROG
OBYR	I	BEGIN YEAR ORIGINAL	PPQUANT	1	PASTPROG
OBMO	I	BEGIN MONTH ORIGINAL	PPQUANT	1	PASTPROG
OEYR	I	END YEAR ORIGINAL	PPQUANT	1	PASTPROG
OEMO	I	END MONTH ORIGINAL	PPQUANT	1	PASTPROG
NZBYR	I	BEGIN YEAR MODIFIED	PPQUANT	1	PASTPROG
NZBMO	I	BEGIN MONTH MODIFIED	PPQUANT	1	PASTPROG
NZEYR	I	END YEAR MODIFIED	PPQUANT	1	PASTPROG
NZEMO	I	END MONTH MODIFIED	PPQUANT	1	PASTPROG
PQUANT	I	PAST PROGRAM MONTHLY QUANTITY	MOQ	2	PASTPROG

OBJECT NAME		MASTER STOCK NUMBER		TECHDATA
MSNAU	L	MASTER STOCK NUMBER	M	TECHDATA
MMC	L	MATERIEL MGMT CODE	M	TECHDATA
FSC	I	FEDERAL SUPPLY CLASSIFICATION	M	TECHDATA
MGR	L	MANAGER DESIGNATOR	M	TECHDATA
NOCOMPUTE	L	NO COMPUTE CODE	M	TECHDATA
PRICE	F	UNIT PRICE	M	TECHDATA
ESC	L	EQUIPMENT SPECIALIST CODE	M	TECHDATA
ITEMNAME	R	ITEM NAME	M	TECHDATA
ERRC	L	EXPENDABILITY REPAIR RECOVERABILITY CODE	M	TECHDATA
ISSUNIT	L	UNIT OF ISSUE	M	TECHDATA
NEWITEM	L	NEW ITEM CODE	M	TECHDATA
CICODE	L	CONTINGENCY/INSURANCE CODE	M	TECHDATA
CILEVEL	I	CONTINGENCY/INSURANCE LEVEL	M	TECHDATA
LTADM	I	LEADTIME ADMINISTRATIVE	M	TECHDATA
LTPROD	I	LEADTIME PRODUCTION	M	TECHDATA
BASERC	I	BASE REPAIR CYCLE	M	TECHDATA
DEPORC	I	DEPOT REPAIR CYCLE	M	TECHDATA
MRLC	L	MAINTENANCE REPAIR LEVEL CODE	M	TECHDATA
BUDPROG	L	BUDGET PROGRAM	M	TECHDATA
WSD	L	WEAPON SYSTEM DESIGNATOR	M	TECHDATA
BUDCODE	L	BUDGET CODE	M	TECHDATA
AASC	L	AIRBORNE ALERT STORAGE CODE	M	TECHDATA
PSC	L	PROGRAM SELECT CODE	M	TECHDATA
PPC	L	PAST PROGRAM PERCENT	M	TECHDATA
PRINTX	L	PRINT EXCEPTION CODE	M	TECHDATA
TDC	L	TDC CODE	M	TECHDATA
TDDR	F	TOTAL DEMAND RATE CURRENT YR	M	TECHDATA
TDDRE	L	TOTAL DEMAND RATE CURRENT YR ESTIMAT CD	M	TECHDATA
TDDR1	F	TOTAL DEMAND RATE 1ST FORECAST YR	M	TECHDATA
TDDRE1	L	TOTAL DEMAND RATE 1ST FORECAST YR ESTIMAT CD	M	TECHDATA
TDDR2	F	TOTAL DEMAND RATE 2ND FORECAST YR	M	TECHDATA
TDDRE2	L	TOTAL DEMAND RATE 2ND FORECAST YR ESTIMAT CD	M	TECHDATA
TDDR3	F	TOTAL DEMAND RATE 3RD FORECAST YR	M	TECHDATA
TDDRE3	L	TOTAL DEMAND RATE 3RD FORECAST YR ESTIMAT CD	M	TECHDATA
BRG	F	BASE REP GEN PERCENT CURRENT	M	TECHDATA
BRG1	F	BASE REP GEN PERCENT 1ST FORECAST	M	TECHDATA
BRG2	F	BASE REP GEN PERCENT 2ND FORECAST	M	TECHDATA
BRG3	F	BASE REP GEN PERCENT 3RD FORECAST	M	TECHDATA
BPROCS	F	BASE PROCESSED PERCENT CURRENT	M	TECHDATA
BPROCS1	F	BASE PROCESSED PERCENT 1ST FORECAST	M	TECHDATA
BPROCS2	F	BASE PROCESSED PERCENT 2ND FORECAST	M	TECHDATA
BPROCS3	F	BASE PROCESSED PERCENT 3RD FORECAST	M	TECHDATA
OBS	L	OBSOLETE CODE	M	TECHDATA
DDC	L	DISPOSAL DEFERRAL CODE	M	TECHDATA
DOFMD	F	DEPOT OEM DEMAND RATE CURRENT	M	TECHDATA
DOFMDE	L	DEPOT OEM DEMAND RATE CURRENT ESTIMATE CODE	M	TECHDATA
DOFMD1	F	DEPOT OEM DEMAND RATE 1ST FORECAST	M	TECHDATA
DOFMDE1	L	DEPOT OEM DEMAND RATE 1ST FORECAST ESTIMATE	M	TECHDATA
DOFMD2	F	DEPOT OEM DEMAND RATE 2ND FORECAST	M	TECHDATA
DOFMDE2	L	DEPOT OEM DEMAND RATE 2ND FORECAST ESTIMATE	M	TECHDATA
DOFMD3	F	DEPOT OEM DEMAND RATE 3RD FORECAST	M	TECHDATA
DOFMDE3	L	DEPOT OEM DEMAND RATE 3RD FORECAST ESTIMATE	M	TECHDATA
BNRTS	F	BASE NRTS PERCENT CURRENT	M	TECHDATA
BNRTSE	L	BASE NRTS PERCENT CURRENT ESTIMATED CODE	M	TECHDATA
BNRTS1	F	BASE NRTS PERCENT 1ST FORECAST	M	TECHDATA
BNRTS2	F	BASE NRTS PERCENT 2ND FORECAST	M	TECHDATA
BNRTS3	F	BASE NRTS PERCENT 3RD FORECAST	M	TECHDATA
IRANWOR	F	IRAN WEAROUT PERCENT CURRENT	M	TECHDATA
IRANWOR1	F	IRAN WEAROUT PERCENT 1ST FORECAST	M	TECHDATA
IRANWOR2	F	IRAN WEAROUT PERCENT 2ND FORECAST	M	TECHDATA
IRANWOR3	F	IRAN WEAROUT PERCENT 3RD FORECAST	M	TECHDATA
PROCUR	L	PROCUREMENT METHOD CODE	M	TECHDATA

OFMBR	F	OFM BASE REPAIR RATE CURRENT	M TECHDATA
OFMBR1	F	OFM BASE REPAIR RATE 1ST FORECAST	M TECHDATA
OFMBR2	F	OFM BASE REPAIR RATE 2ND FORECAST	M TECHDATA
OFMBR3	F	OFM BASE REPAIR RATE 3RD FORECAST	M TECHDATA
BCCND	F	BASE CONDEMNATIONS PERCENT CURRENT	M TECHDATA
BCONDE	L	BASE CONDEMNATIONS PERCENT CURRENT ESTIMATE	M TECHDATA
BCOND1	F	BASE CONDEMNATIONS PERCENT 1ST FORECAST	M TECHDATA
BCOND2	F	BASE CONDEMNATIONS PERCENT 2ND FORECAST	M TECHDATA
BCOND3	F	BASE CONDEMNATIONS PERCENT 3RD FORECAST	M TECHDATA
EOHWOR	F	ENGINE OVERHAUL WEAROUT PERCENT CURRENT	M TECHDATA
EOHWOR1	F	ENGINE OVERHAUL WEAROUT PERCENT 1ST FORECAST	M TECHDATA
EOHWOR2	F	ENGINE OVERHAUL WEAROUT PERCENT 2ND FORECAST	M TECHDATA
EOHWOR3	F	ENGINE OVERHAUL WEAROUT PERCENT 3RD FORECAST	M TECHDATA
WOR	F	WEAROUT RATE CURRENT	M TECHDATA
WORE	L	WEAROUT RATE CURRENT ESTIMATED CODE	M TECHDATA
WOR1	F	WEAROUT RATE 1ST FORECAST	M TECHDATA
WORE1	L	WEAROUT RATE 1ST FORECAST ESTIMATED CODE	M TECHDATA
WOR2	F	WEAROUT RATE 2ND FORECAST	M TECHDATA
WORE2	L	WEAROUT RATE 2ND FORECAST ESTIMATED CODE	M TECHDATA
WOR3	F	WEAROUT RATE 3RD FORECAST	M TECHDATA
WORE3	L	WEAROUT RATE 3RD FORECAST ESTIMATED CODE	M TECHDATA
DOHCOND	F	DEPOT OVERHAUL CONDEMN PERCENT CURRENT	M TECHDATA
DOHCONDE	L	DEPOT OVERHAUL CONDEMN PERCENT CURRENT ESTIM	M TECHDATA
DOHCOND1	F	DEPOT OVERHAUL CONDEMN PERCENT 1ST FORECAST	M TECHDATA
DOHCOND2	F	DEPOT OVERHAUL CONDEMN PERCENT 2ND FORECAST	M TECHDATA
DOHCOND3	F	DEPOT OVERHAUL CONDEMN PERCENT 3RD FORECAST	M TECHDATA
MRSWOR	F	MRS WEAROUT PERCENT CURRENT	M TECHDATA
MRSWOR1	F	MRS WEAROUT PERCENT 1ST FORECAST	M TECHDATA
MRSWOR2	F	MRS WEAROUT PERCENT 2ND FORECAST	M TECHDATA
MRSWOR3	F	MRS WEAROUT PERCENT 3RD FORECAST	M TECHDATA
TODRZ	F	TOTAL OFM DEMAND RATE	M TECHDATA
TODRZE	L	TOTAL OFM DEMAND RATE ESTIMATED CODE	M TECHDATA
DOFMDZ	F	DEPOT OFM DEMAND RATE	M TECHDATA
DOFMDZE	L	DEPOT OFM DEMAND RATE ESTIMATED CODE	M TECHDATA
WORZ	F	WEAROUT RATE	M TECHDATA
WORZE	L	WEAROUT RATE ESTIMATED CODE	M TECHDATA
OFMBRZ	F	OFM BASE REPAIR RATE	M TECHDATA
ZRGYRS	I	NUMBER YEARS ZERO REP GENS	M TECHDATA
BCONDZ	F	BASE CONDEMNATION PERCENT	M TECHDATA
BCONDZE	L	BASE CONDEMNATION PERCENT ESTIMATED CODE	M TECHDATA
DOHCONDZ	F	DEPOT OVERHAUL CONDEMNATION PERCENT	M TECHDATA
DOHCONDZE	L	DEPOT OVERHAUL CONDEMN PERCENT ESTIMATED	M TECHDATA
BNRTSZ	F	BASE NRTS PERCENT	M TECHDATA
BNRTSZE	L	BASE NRTS PERCENT ESTIMATED CODE	M TECHDATA
ASNCTR	I	TOTAL NUMBER CURRENT STOCK NUMBERS	M TECHDATA
OBSCTR	I	TOTAL NUMBER CURRENT STOCK NUMBERS OBSOLETE	M TECHDATA
BPBRG	I	BASE PERIOD BASE REPARABLE GENERATION	M TECHDATA
BPBCOND	I	BASE PERIOD BASE CONDEMNATIONS	M TECHDATA
BPBRTS	I	BASE PERIOD BASE RTS	M TECHDATA
BPBNRTS	I	BASE PERIOD BASE NRTS	M TECHDATA
BPDREP	I	BASE PERIOD DEPOT REPAIRED	M TECHDATA
BPDOHC	I	BASE PERIOD DEPOT OVERHAUL CONDEMNATION	M TECHDATA
BPPAST	I	BASE PERIOD PAST PROGRAM	M TECHDATA
RRG6	I	RIAR REPARABLE GENERATION 6 MONTHS	M TECHDATA
RRG12	I	RIAR REPARABLE GENERATION 12 MONTHS	M TECHDATA
RCOND6	I	RIAR CONDEMNATION 6 MONTHS	M TECHDATA
RCOND12	I	RIAR CONDEMNATION 12 MONTHS	M TECHDATA
RRTS6	I	RIAR RTS 6 MONTHS	M TECHDATA
RRTS12	I	RIAR RTS 12 MONTHS	M TECHDATA
RNRTS6	I	RIAR NRTS 6 MONTHS	M TECHDATA
RNRTS12	I	RIAR NRTS 12 MONTHS	M TECHDATA
RDCOND6	I	RIAR DEPOT CONDEMNATION 6 MONTHS	M TECHDATA
RDCOND12	I	RIAR DEPOT CONDEMNATION 12 MONTHS	M TECHDATA
SLBOFM	I	STOCK LEVEL BASE OFM	M TECHDATA

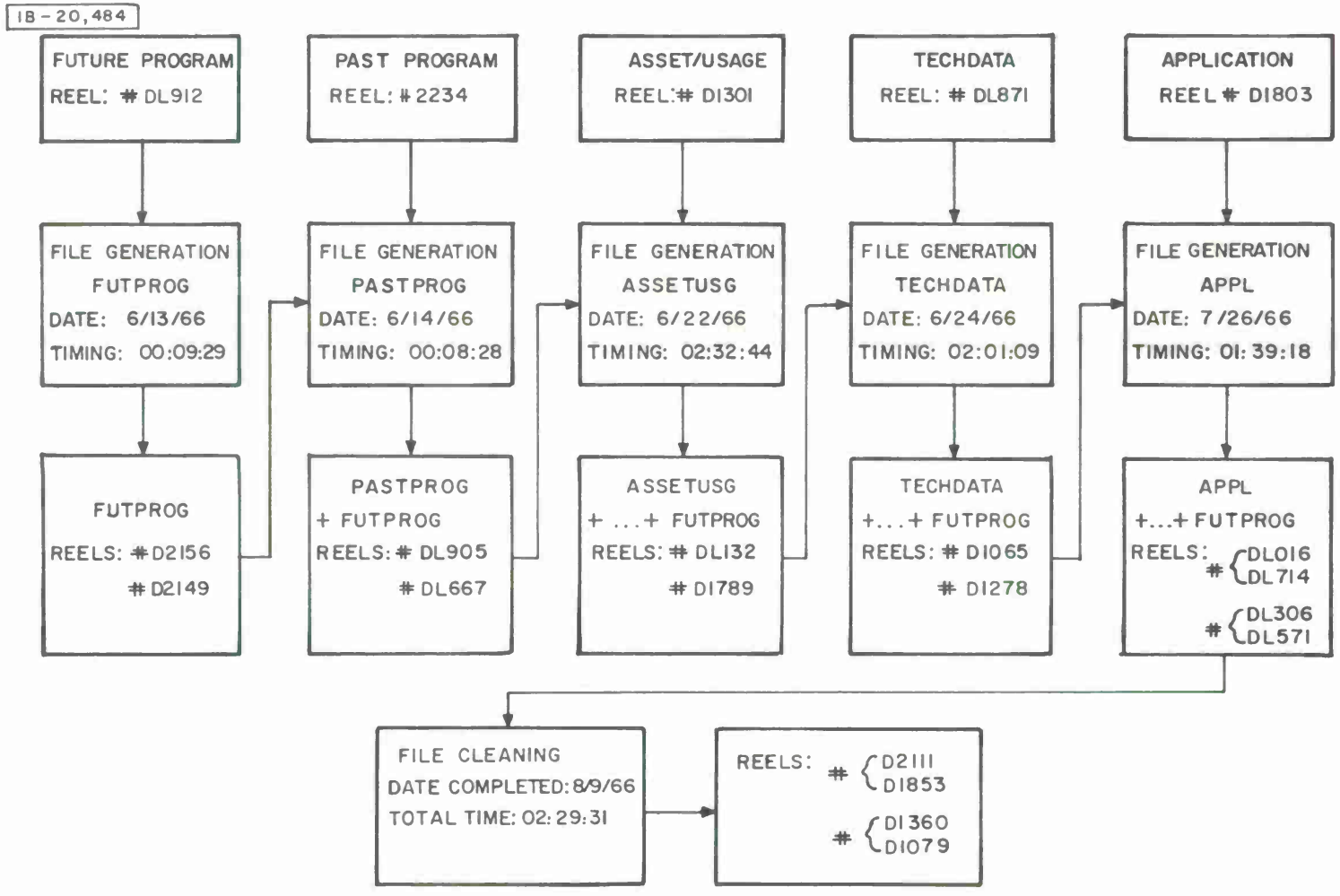
SLDCFM	I	STOCK LEVEL DEPOT OFM	M TECHDATA
SLAA	I	STOCK LEVEL AIRBORNE ALERT	M TECHDATA
SLOH	I	STOCK LEVEL OVERHAUL	M TECHDATA
DFSL	I	DEPOT O/H FLOATING STK LEVEL	M TECHDATA
STDHRS	I	STANDARD DIRECT PRODUCTION HOURS	M TECHDATA
NETBUY	I	NET BUY QUANTITY	M TECHDATA
NJRIRAN	F	NON-JOB ROUTED PERCENT IRAN	M TECHDATA
NJREOH	F	NON-JOB ROUTED PERCENT ENGINE OVERHAUL	M TECHDATA
NJRMRS	F	NON-JOB ROUTED PERCENT MRS	M TECHDATA
IRANNJRR	F	IRAN NON JOB ROUTED REPLACE RATE CURRENT	M TECHDATA
IRANNJRR1	F	IRAN NON JOB ROUTED REPLACE RATE 1ST FORECST	M TECHDATA
IRANNJRR2	F	IRAN NON JOB ROUTED REPLACE RATE 2ND FORECST	M TECHDATA
IRANNJRR3	F	IRAN NON JOB ROUTED REPLACE RATE 3RD FORECST	M TECHDATA
EOHNJRR	F	EOH NON JOB ROUTED REPLACE RATE CURRENT	M TECHDATA
EOHNJRR1	F	EOH NON JOB ROUTED REPLACE RATE 1ST FORECST	M TECHDATA
EOHNJRR2	F	EOH NON JOB ROUTED REPLACE RATE 2ND FORECST	M TECHDATA
EOHNJRR3	F	EOH NON JOB ROUTED REPLACE RATE 3RD FORECST	M TECHDATA
MRSNJRR	F	MRS NON JOB ROUTED REPLACE RATE CURRENT	M TECHDATA
MRSNJRR1	F	MRS NON JOB ROUTED REPLACE RATE 1ST FORCAST	M TECHDATA
MRSNJRR2	F	MRS NON JOB ROUTED REPLACE RATE 2ND FORCAST	M TECHDATA
MRSNJRR3	F	MRS NON JOB ROUTED REPLACE RATE 3RD FORCAST	M TECHDATA
ACUTYR	I	START DATE YEAR	M TECHDATA
ACUTMO	I	START DATE MONTH	M TECHDATA
NBSL1	I	NEGOTIATED BASE STK LEVEL 1ST QUARTER	M TECHDATA
NBSL2	I	NEGOTIATED BASE STK LEVEL 2ND QUARTER	M TECHDATA
NBSL3	I	NEGOTIATED BASE STK LEVEL 3RD QUARTER	M TECHDATA
NBSL4	I	NEGOTIATED BASE STK LEVEL 4TH QUARTER	M TECHDATA
NBSL5	I	NEGOTIATED BASE STK LEVEL 5TH QUARTER	M TECHDATA
NBSL6	I	NEGOTIATED BASE STK LEVEL 6TH QUARTER	M TECHDATA
NBSL7	I	NEGOTIATED BASE STK LEVEL 7TH QUARTER	M TECHDATA
NBSL8	I	NEGOTIATED BASESTK LEVEL 8TH QUARTER	M TECHDATA
NBSL9	I	NEGOTIATED BASE STK LEVEL 9TH QUARTER	M TECHDATA
NBSL10	I	NEGOTIATED BASE STK LEVEL 10TH QUARTER	M TECHDATA
NBLS11	I	NEGOTIATED BASE STK LEVEL 11TH QUARTER	M TECHDATA
NBLS12	I	NEGOTIATED BASE STK LEVEL 12TH QUARTER	M TECHDATA
NBLS13	I	NEGOTIATED BASE STK LEVEL 13TH QUARTER	M TECHDATA
TABRG1	F	TEMPORARY STORAGE BUFFER	M TECHDATA
TABRG2	F	TEMPORARY STORAGE BUFFER	M TECHDATA
TABRG3	F	TEMPORARY STORAGE BUFFER	M TECHDATA
TABRTS1	F	TEMPORARY STORAGE BUFFER	M TECHDATA
TABRTS2	F	TEMPORARY STORAGE BUFFER	M TECHDATA
TABRTS3	F	TEMPORARY STORAGE BUFFER	M TECHDATA
TABCOND1	F	TEMPORARY STORAGE BUFFER	M TECHDATA
TABCOND2	F	TEMPORARY STORAGE BUFFER	M TECHDATA
TABCOND3	F	TEMPORARY STORAGE BUFFER	M TECHDATA
CYR	I	CURRENT YEAR TEMPORARY	M TECHDATA
CMO	I	CURRENT MONTH TEMPORARY	M TECHDATA
SYR	I	START YEAR TEMPORARY	M TECHDATA
SMO	I	START MONTH TEMPORARY	M TECHDATA
SPAN	I	NUMBER OF YEARS TEMPORARY	M TECHDATA

M=MAIN
1=REPEATING GROUP LEVEL 1
2=REPEATING GROUP LEVEL 2

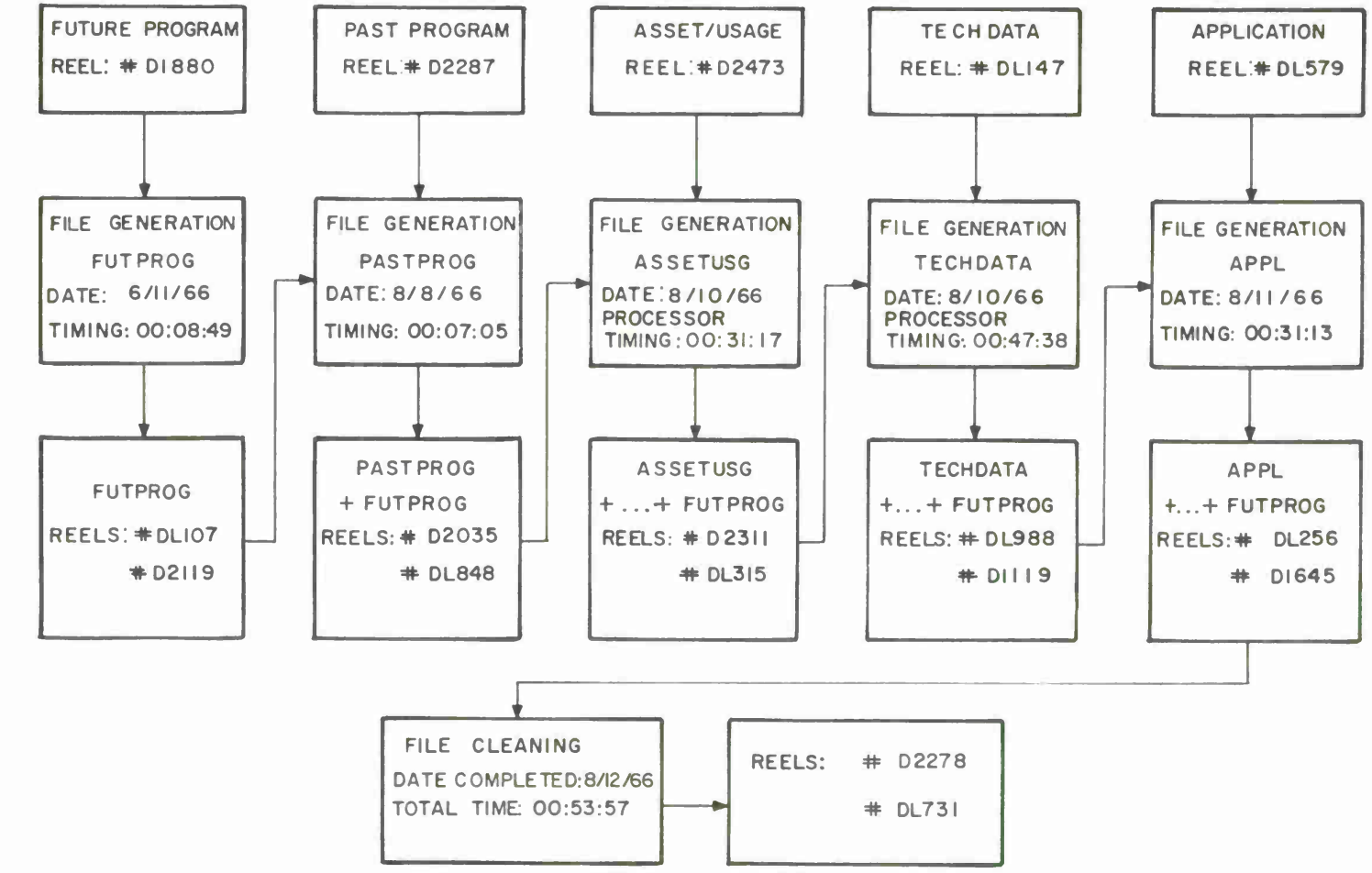
F=FLOATING
I=INTEGER
L=LOGICAL
R=RAW

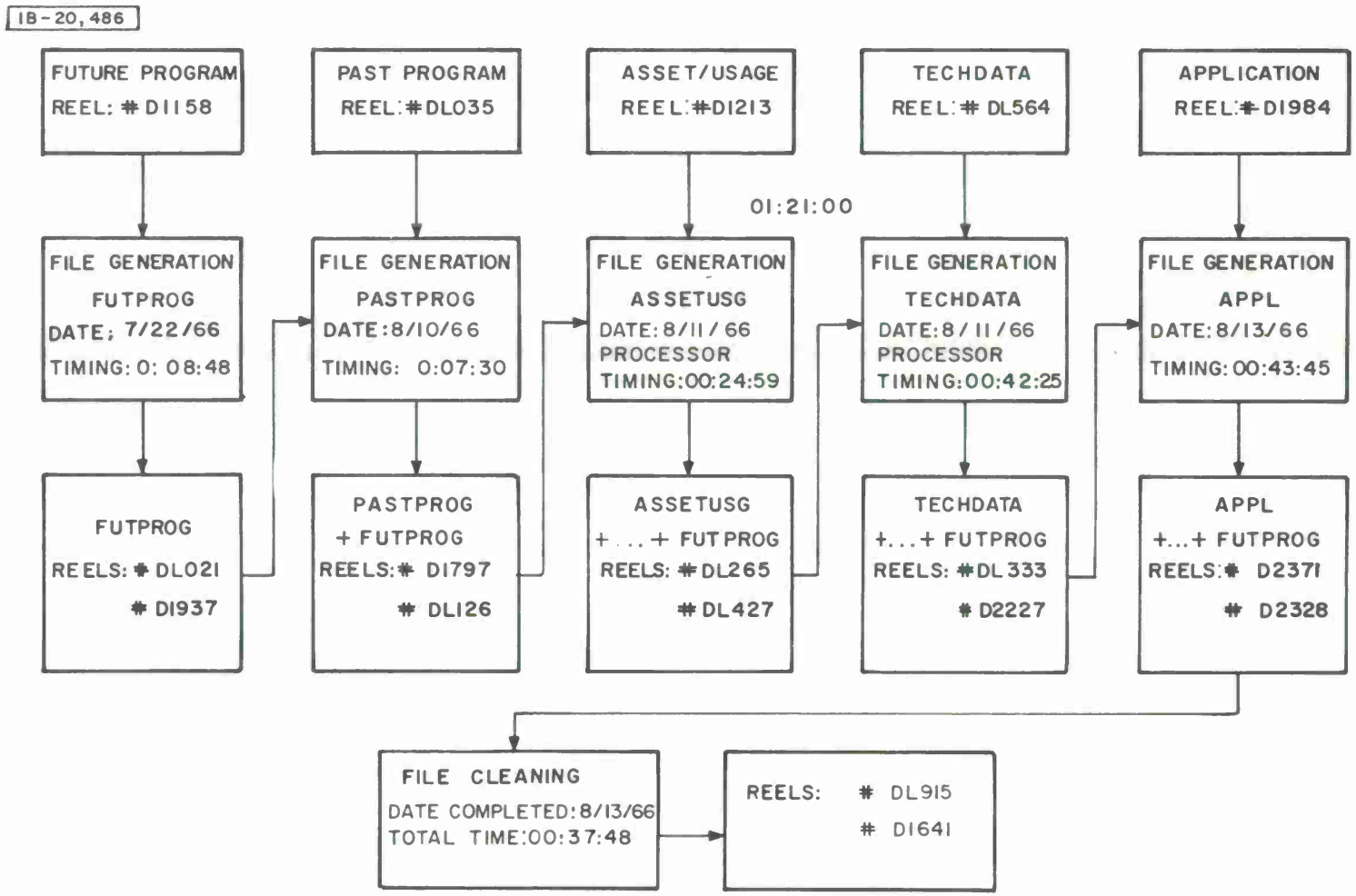
APPENDIX V

PROCEDURE FOR FILE GENERATION

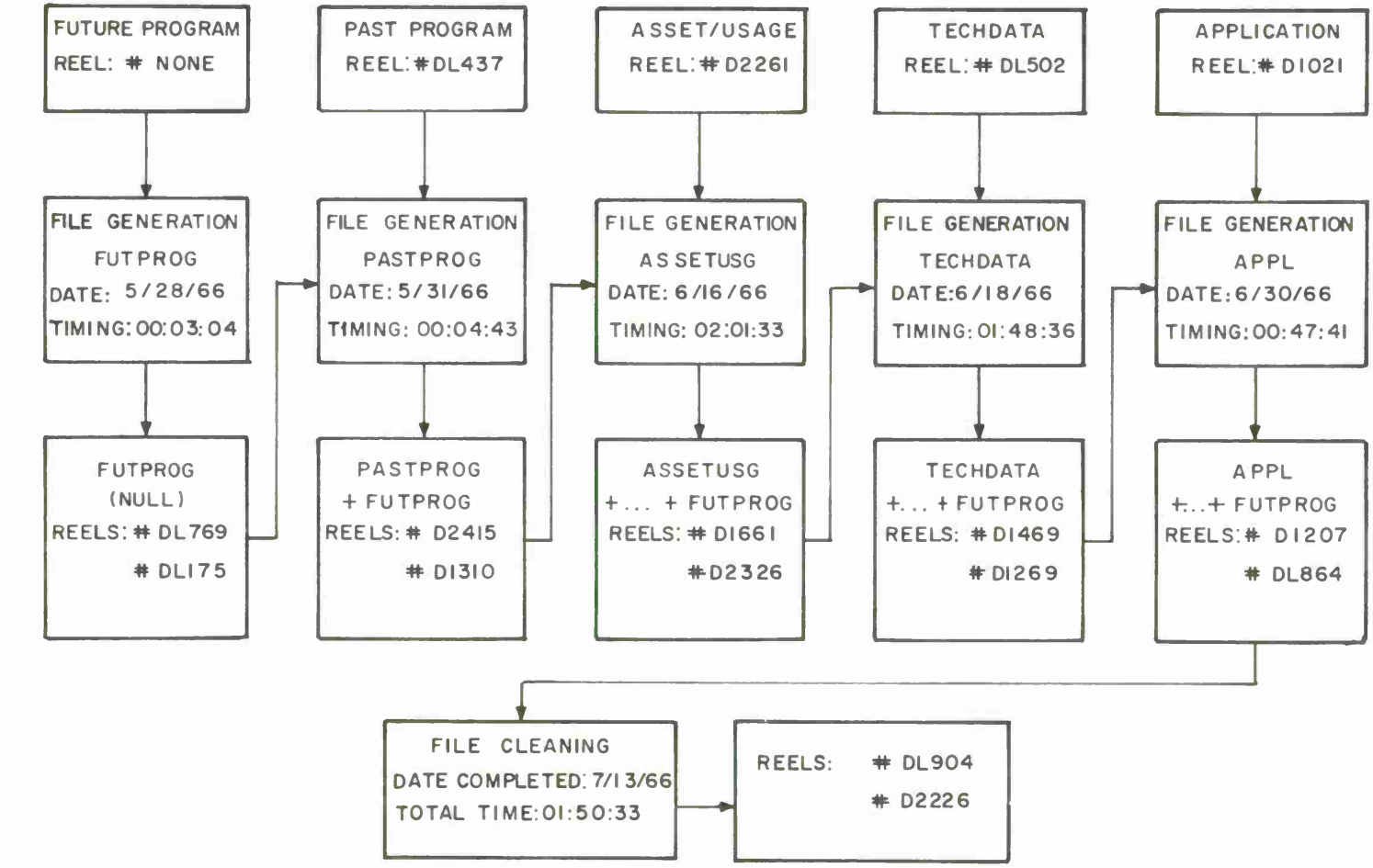


IB-20,485

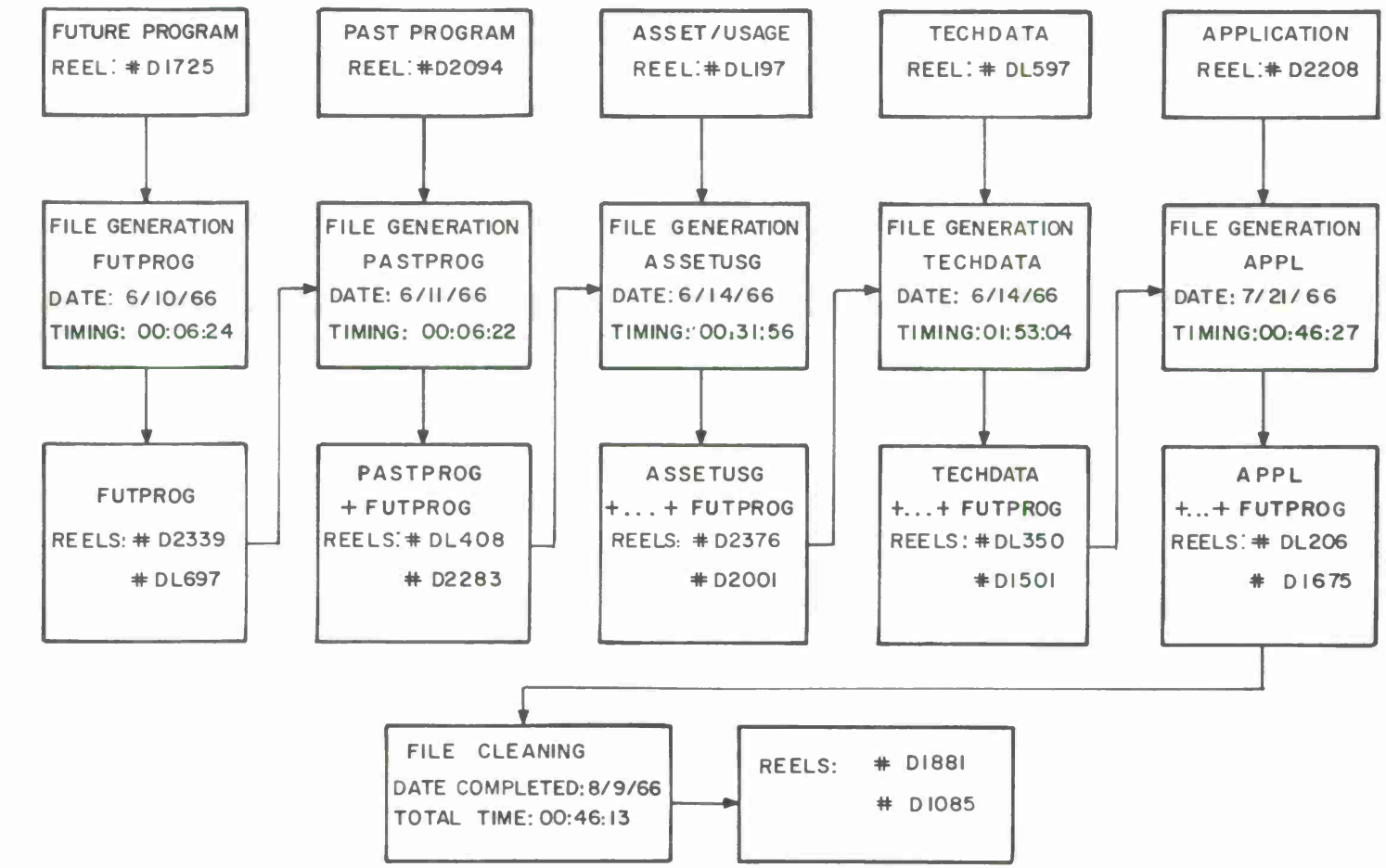

FUTURE PROGRAM
REEL: # DI880
PAST PROGRAM
REEL: # D2287
ASSET/USAGE
REEL: # D2473
TECH DATA
REEL: # DLI47
APPLICATION
REEL: # DL579
FILE GENERATION
FUTPROG
DATE: 6/11/66
TIMING: 00:08:49
FILE GENERATION
PASTPROG
DATE: 8/8/66
TIMING: 00:07:05
FILE GENERATION
ASSETUSG
DATE: 8/10/66
PROCESSOR
TIMING: 00:31:17
FILE GENERATION
TECHDATA
DATE: 8/10/66
PROCESSOR
TIMING: 00:47:38
FILE GENERATION
APPL
DATE: 8/11/66
TIMING: 00:31:13
FUTPROG
REELS: # DLI07
D2119
PASTPROG
+ FUTPROG
REELS: # D2035
DL848
ASSETUSG
+...+ FUTPROG
REELS: # D2311
DL315
TECHDATA
+...+ FUTPROG
REELS: # DL988
DIII9
APPL
+...+ FUTPROG
REELS: # DL256
DI645
FILE CLEANING
DATE COMPLETED: 8/12/66
TOTAL TIME: 00:53:57
REELS: # D2278
DL731

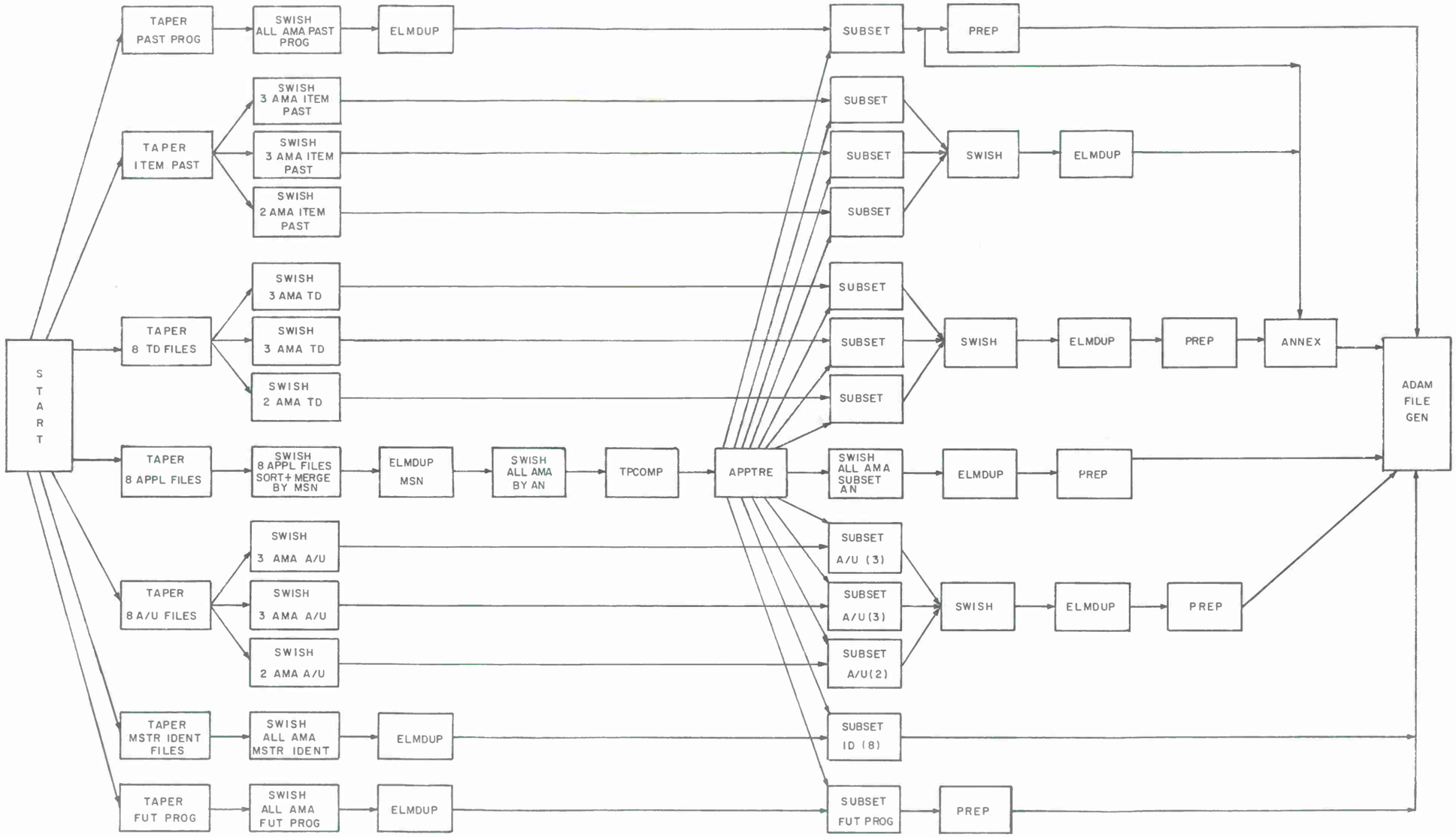

IB-20,486
FUTURE PROGRAM
REEL: # D1158
PAST PROGRAM
REEL: # DL035
ASSET/USAGE
REEL: # D1213
TECHDATA
REEL: # DL564
APPLICATION
REEL: # D1984
01:21:00
FILE GENERATION
FUTPROG
DATE; 7/22/66
TIMING: 0: 08:48
FILE GENERATION
PASTPROG
DATE: 8/10/66
TIMING: 0:07:30
FILE GENERATION
ASSETUSG
DATE: 8/11/66
PROCESSOR
TIMING: 00:24:59
FILE GENERATION
TECHDATA
DATE: 8/11/66
PROCESSOR
TIMING: 00:42:25
FILE GENERATION
APPL
DATE: 8/13/66
TIMING: 00:43:45
FUTPROG
REELS: # DL021
D1937
PASTPROG
+ FUTPROG
REELS: # D1797
DL126
ASSETUSG
+ ... + FUTPROG
REELS: # DL265
DL427
TECHDATA
+ ... + FUTPROG
REELS: # DL333
D2227
APPL
+ ... + FUTPROG
REELS: # D2371
D2328
FILE CLEANING
DATE COMPLETED: 8/13/66
TOTAL TIME: 00:37:48
REELS: # DL915
D1641

IB-20,487

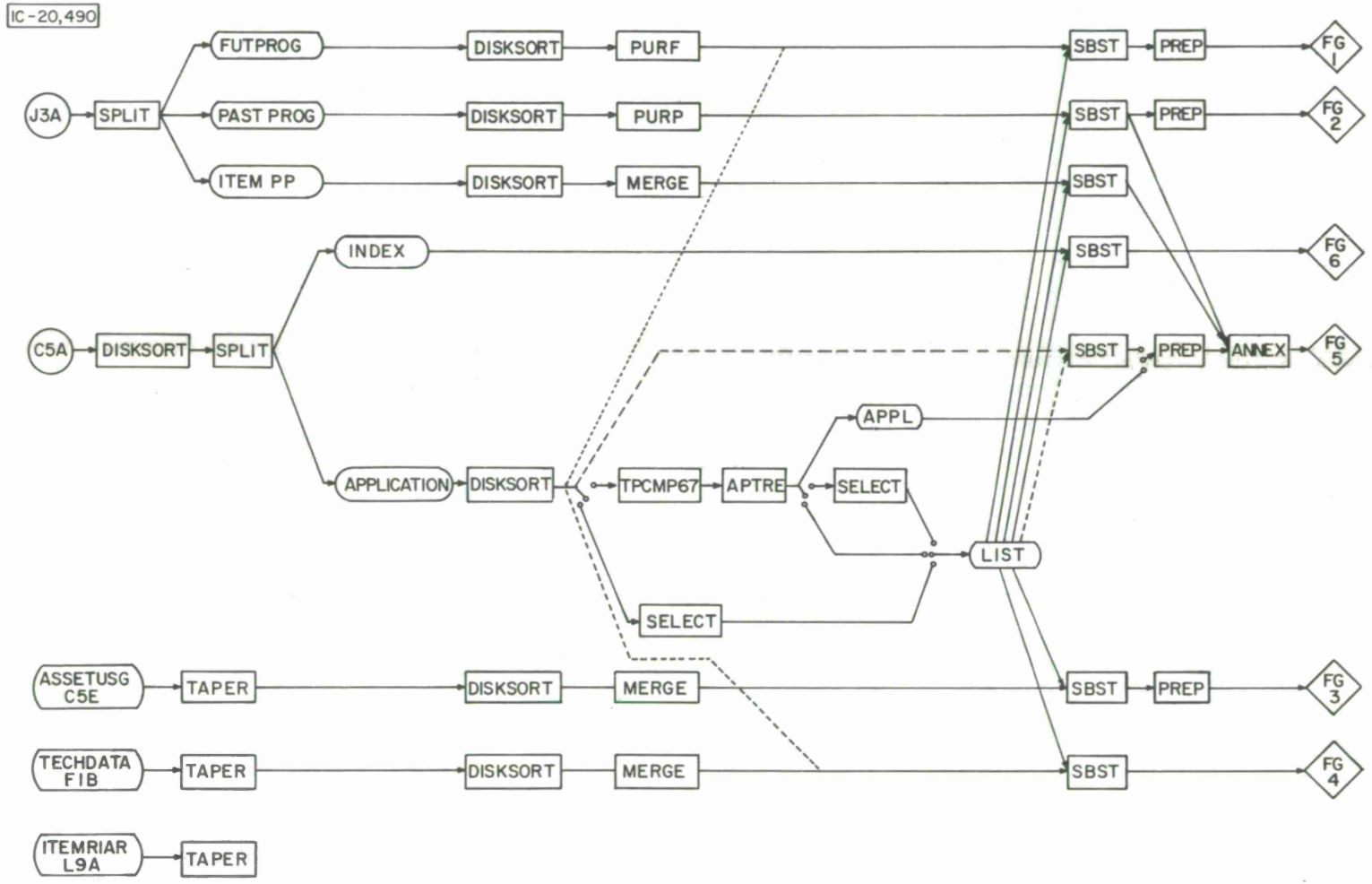

FUTURE PROGRAM
REEL: # NONE
PAST PROGRAM
REEL: #DL437
ASSET/USAGE
REEL: # D2261
TECHDATA
REEL: # DL502
APPLICATION
REEL: # D1021
FILE GENERATION
FUTPROG
DATE: 5/28/66
TIMING: 00:03:04
FILE GENERATION
PASTPROG
DATE: 5/31/66
TIMING: 00:04:43
FILE GENERATION
ASSETUSG
DATE: 6/16/66
TIMING: 02:01:33
FILE GENERATION
TECHDATA
DATE: 6/18/66
TIMING: 01:48:36
FILE GENERATION
APPL
DATE: 6/30/66
TIMING: 00:47:41
FUTPROG
(NULL)
REELS: # DL769
DL175
PASTPROG
+ FUTPROG
REELS: # D2415
D1310
ASSETUSG
+... + FUTPROG
REELS: # D1661
D2326
TECHDATA
+... + FUTPROG
REELS: # D1469
D1269
APPL
+...+ FUTPROG
REELS: # D1207
DL864
FILE CLEANING
DATE COMPLETED: 7/13/66
TOTAL TIME: 01:50:33
REELS: # DL904
D2226

IB-20,488


FUTURE PROGRAM
REEL: # D1725
PAST PROGRAM
REEL: #D2094
ASSET/USAGE
REEL: # DL197
TECHDATA
REEL: # DL597
APPLICATION
REEL: # D2208
FILE GENERATION
FUTPROG
DATE: 6/10/66
TIMING: 00:06:24
FILE GENERATION
PASTPROG
DATE: 6/11/66
TIMING: 00:06:22
FILE GENERATION
ASSETUSG
DATE: 6/14/66
TIMING: 00:31:56
FILE GENERATION
TECHDATA
DATE: 6/14/66
TIMING: 01:53:04
FILE GENERATION
APPL
DATE: 7/21/66
TIMING: 00:46:27
FUTPROG
REELS: # D2339
DL697
PASTPROG
+ FUTPROG
REELS: # DL408
D2283
ASSETUSG
+... + FUTPROG
REELS: # D2376
D2001
TECHDATA
+... + FUTPROG
REELS: # DL350
D1501
APPL
+... + FUTPROG
REELS: # DL206
D1675
FILE CLEANING
DATE COMPLETED: 8/9/66
TOTAL TIME: 00:46:13
REELS: # D1881
D1085


START
TAPER PAST PROG
SWISH ALL AMA PAST PROG
ELMDUP
TAPER ITEM PAST
SWISH 3 AMA ITEM PAST
SWISH 3 AMA ITEM PAST
SWISH 2 AMA ITEM PAST
TAPER 8 TD FILES
SWISH 3 AMA TD
SWISH 3 AMA TD
SWISH 2 AMA TD
TAPER 8 APPL FILES
SWISH 8 APPL FILES SORT+ MERGE BY MSN
ELMDUP MSN
SWISH ALL AMA BY AN
TPCOMP
APPTRE
TAPER 8 A/U FILES
SWISH 3 AMA A/U
SWISH 3 AMA A/U
SWISH 2 AMA A/U
TAPER MSTR IDENT FILES
SWISH ALL AMA MSTR IDENT
ELMDUP
TAPER FUT PROG
SWISH ALL AMA FUT PROG
ELMDUP
SUBSET
PREP
SUBSET
SUBSET
SUBSET
SWISH
ELMDUP
SUBSET
SUBSET
SUBSET
SWISH
ELMDUP
PREP
ANNEX
SWISH ALL AMA SUBSET AN
ELMDUP
PREP
SUBSET A/U (3)
SUBSET A/U(3)
SUBSET A/U(2)
SWISH
ELMDUP
PREP
SUBSET ID (8)
SUBSET FUT PROG
PREP
ADAM FILE GEN

ID - 20,491


IC-20,490
J3A
SPLIT
FUTPROG
PAST PROG
ITEM PP
DISKSORT
DISKSORT
DISKSORT
PURF
PURP
MERGE
C5A
DISKSORT
SPLIT
INDEX
APPLICATION
DISKSORT
TPCMP67
APTRE
APPL
SELECT
SELECT
LIST
SBST
SBST
SBST
SBST
SBST
PREP
PREP
PREP
ANNEX
FG 1
FG 2
FG 6
FG 5
ASSETUSG C5E
TAPER
DISKSORT
MERGE
SBST
PREP
FG 3
TECHDATA F1B
TAPER
DISKSORT
MERGE
SBST
FG 4
ITEMRIAR L9A
TAPER

REFERENCES

1. A.C. Foreman, "Advanced Data Management Experiment," IEEE Transactions in Aerospace and Electronic Systems, AES-2, January 1966, 115-120.

2. F. Cataldo and B. F. Char, "Data Base Analysis," The MITRE Corporation, Bedford, Massachusetts, MTR-55, 12 November 1965.

3. B. F. Char, "Data Base Analysis," The MITRE Corporation, Bedford, Massachusetts, MTR-55, Supp. 1, 27 December 1965.

4. A. J. Nickelson, "Data Base Analysis," The MITRE Corporation, Bedford, Massachusetts, MTR-55, Supp. 2, 4 February 1966.

5. A. J. Nickelson, "Data Base Analysis," The MITRE Corporation, Bedford, Massachusetts, MTR-55, Supp. 3, August 1966.

6. B. F. Char, "General Program Design for ADAM-Based AFLC DO41 System," The MITRE Corporation, Bedford, Massachuetts, MTR-77, 3 January 1966.

7. O. W. Beebe, "Factor Computations for the ADAM-Based AFLC DO41 System," The MITRE Corporation, Bedford, Massachusetts, MTR-137, Rev. 1, 11 July 1966.

8. J. C. Penney, "Requirements Computation for the ADAM-Based AFLC DO41 System," The MITRE Corporation, Bedford, Massachusetts, MTR-168, Rev. 1, 6 July 1966.

9. O. W. Beebe, "Requirements Inventory Analysis Report (RIAR), Computations for the ADAM-Based AFLC DO41 Systems," The MITRE Corporation, Bedford, Massachusetts, MTR-238, 11 July 1966.

10. S. Bramson, "AFLC Remote Equipment Checkout program," The MITRE Corporation, Bedford, Massachusetts, MTR-243, 11 July 1966.

11. I. Beilin, "MERGE," The MITRE Corporation, Bedford, Massachusetts, MTR-271, August 1966.

12. O.W. Beebe and J. C. Penney, "Implementation of ADAM-AFLC Experiment--Phase II," The MITRE Corporation, Bedford, Massachusetts, MTR-262, 15 August 1966.

REFERENCES (Concl'd)

13. O. W. Beebe, B. F. Char, and J. C. Penney, "Implementation of ADAM-AFLC Experiment--Phase I," The MITRE Corporation, Bedford, Massachusetts, MTR-109, 24 January 1966.

14. J. C. Penney, "ADAM/AFLC Demonstration," The MITRE Corporation, Bedford, Massachusetts, MTR-272, August 1966.

15. ADAM Project, "Guide to the ADAM System," The MITRE Corporation, Bedford, Massachusetts, MTR-268, 8 August 1966.

16. S. Bramson, "AFLCON, PPCONV, APOWER Routines for Project 5030," The MITRE Corporation, Bedford, Massachusetts, MTR-257, 25 July 1966.

17. T. L. Connors, "ADAM: A Generalized Data Management System," The MITRE Corporation, Bedford, Massachusetts, MTP-29, August 1966.

18. R. Conrod, "MCP Modifications to Support Project 503," The MITRE Corporation, MTR-315, 28 October 1966.

BIBLIOGRAPHY

Cataldo, F., "Decision on Equipment Augmentation," The MITRE Corporation, Bedford, Massachusetts, MTR-62, 15 November 1965.

Computer Associates, Inc., Wakefield, Massachusetts, and EDS, AFSC, USAF, Hanscom Air Force Base, Bedford, Massachusetts, "Advanced Planning Developments: A Survey," ESD-TR-65-171, February 1965.

Nickelson, A. J., "Routines for Processing AFLC Data," The MITRE Corporation, Bedford, Massachusetts, MTR-270, August 1966.

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
The MITRE Corporation
Bedford, Massachusetts

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE
Final Report-Joint AFLC/ESD/MITRE Advanced Data Management (ADAM) Experiment

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
N/A

5. AUTHOR(S) *(Last name, first name, initial)*
Char, Beverly F.
Foreman, Alling C.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
February 1967	123	20

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
AF - 19(628)5128	ESD-TR-66-330
b. PROJECT NO. 503F	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* MTR-285
d.	

10. AVAILABILITY/LIMITATION NOTICES
Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
N/A	Deputy for Engineering and Technology, Computer Programming Division, Electronic Systems Division, L.G.Hanscom Field, Bedford, Massachusetts.

13. ABSTRACT

This final report describes the components of the Joint AFLC/ESD/MITRE Advanced Data Management Experiment (ADAM) and the process of implementation. The objective of the experiment was to determine the applicability of Generalized Data Management Systems such as ADAM to management information problems as found in AFLC. Observations concerning this applicability are given from two user viewpoints: programmer-user and the application or mission-oriented user.

DD FORM 1 JAN 64 1473

UNCLASSIFIED

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Data Management						
On-line						
Retrieval						
Query						
ADAM						
Logistics						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2*a*. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2*b*. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7*a*. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7*b*. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8*a*. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8*b*, 8*c*, & 8*d*. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9*a*. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9*b*. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.

GPO 886-551